Distributing Applications with the LabVIEW Application Builder

Publish Date: aug 23, 2013 | 140 Ratings | **2.94** out of 5 | Print | 23 Customer Reviews | Submit your review

Overview

This article is part of a series on software engineering practices and tools for large application development in LabVIEW.

Click here to view the list of other articles

Software developers rarely intend for a finished application to be used only on the development computer. To address this issue, National Instruments provides the LabVIEW Application Builder, a user-friendly tool that enables LabVIEW users to configure build specifications:

- **Stand-alone applications** that can be distributed to users who have the LabVIEW Run-time Engine.
- **Packed project libraries** that package multiple files in one location with a project library as the root file.
- **Shared libraries** that call LabVIEW code from text-based programming languages, such as LabWindows/CVI, Microsoft Visual C++, and Microsoft Visual Basic.
- **Source distributions** that package LabVIEW source files, which can be distributed to other LabVIEW developers.
- **Installers** that install stand-alone applications and shared libraries created for the Windows platform.
- **Zip files** that are useful for distributing a large amount of code as a single, portable file.

Using build specifications differs significantly depending on the version of LabVIEW you use. Refer to the *LabVIEW Help* for the most detailed and up-to-date information about the Application Builder. You can access the *LabVIEW Help* by selecting **Help»LabVIEW Help** in LabVIEW.

The LabVIEW Professional Development System includes the Application Builder. If you use the LabVIEW Base Package or Full Development System, you can purchase the Application Builder separately.

Note: You must have LabVIEW 8 or higher to complete these tutorials. Some text may differ in the software depending on which version of LabVIEW you use.

Table of Contents

Bookmark & Share
Share

Downloads
Attachments:
tutorialfiles.zip

Ratings
Rate this document
Select a Rating

Answered Your Question?
Yes No

Submit

Building Applications with the Application Builder

The LabVIEW Application Builder leverages the organization provided by the LabVIEW Project, which organizes and manages all of the files associated with an application. These files include VIs, project libraries, documentation, data files, hardware configuration and more. The Application Builder creates applications from user-specified files in a LabVIEW project, and individual build settings are saved in the project.

The following examples use existing VIs to step through the tasks for configuring build specifications in LabVIEW. Download the attached VIs to complete each example. You must have a project open and saved to configure a build specification.

1. Preparing the LabVIEW Project

1. Download and unzip the file `tutorial_files.zip` attached to this document.
2. Select **File»New Project** to create a project.
3. Right-click **My Computer** and select **Add»Folder** from the shortcut menu to add a directory of files to the project as a folder. Navigate to the files from `tutorial_files.zip`.
4. Click the **Current Folder** button to add the directory to the project. The **Application Builder** folder appears under **My Computer** in the **Project Explorer** window.
5. Save the project as `TutorialProject` in a new directory named `AB Tutorial`. Use this project to complete the examples.

Note If you previously hid **Build Specifications** in the **Project Explorer** window, you must display the item again to access it. To view **Build Specifications**, click the **Filter Project View** button and select **Build Specifications**.

2. Stand-Alone Applications

Back to Top

You can develop stand-alone applications (executables) that only require the LabVIEW Run-Time Engine to run on systems without LabVIEW installed. Complete the following steps in each task to configure the application build specification and build and run the application. Refer to the *LabVIEW Help* for the most detailed and up-to-date information about building stand-alone applications. You can access the *LabVIEW Help* by selecting **Help»LabVIEW Help** in LabVIEW.

Configuring the Build Specification

1. From the **Project Explorer** window, right-click **Build Specifications** and select **New»Application** from the shortcut menu to display the **Application Properties** dialog box.
2. Complete the following steps on the **Information** page of the dialog box.
    1. Enter `Tutorial-EXE` in the **Build specification name** text box. The text box contains the name of the application build specification that appears under **Build Specifications** in the **Project Explorer** window.
    2. Enter `Tutorial.exe` in the **Target filename** text box. The text box contains the filename on disk for the executable file.
    3. Click the **Browse** button next to the **Destination directory** text box and navigate to the `AB Tutorial` directory. Create a subdirectory named `App`. Double-click the `App` directory and click the **Current Folder** button to select the directory. When you build the application, the executable file appears in this directory.
3. From the **Category** list on the left, click **Source Files** to display the **Source Files** page, which contains settings you can use to specify which files are included and how they are included in the application. Complete the following steps on the dialog box page.
    1. From the **Project Files** tree, click the **Tutorial Files** folder to expand it. Select **AB Tutorial GUI.vi**. Click the **Add Item** arrow button next to the **Startup VIs** listbox to add the VI to the listbox. VIs you add to the **Startup VIs** listbox open and run when you run the application.

2. From the **Project Files** tree, select **AB Tutorial MeanMedianMode.vi**. Click the **Add Item** arrow button next to the **Always Included** listbox to add the VI to the listbox. Files you add to the **Always Included** listbox are in the application, but the application calls them dynamically as needed.
3. Repeat the previous step to add the following files to the **Always Included** listbox: `AB Tutorial StndDev.vi`, `AB Tutorial Histogram.vi`, and `datafile.txt`. Do not add the other files, which are for use in other examples.

Note If you move a VI to the **Always Included** listbox on the **Source Files** page, but you choose a destination in the **Source File Settings** page that is not the same destination as the built application, the Application Builder moves all dependencies of the VI to the destination you designate, rather than keep the dependencies in the built application. If two or more VIs that are always included or top-level VIs call a VI and try to move it to two different locations, the Application Builder moves the VI and all subVIs to the built application. To make a VI that is specified as **Include if referenced** move to a new location, you must specify the VI as **Always Included** on the **Source Files** page.

4. From the **Category** list, select the other dialog box pages to explore the settings you can configure in a build specification for an application. For this exercise, do not change the settings on these dialog box pages.
5. From the **Category** list, click **Preview** to display the **Preview** page. Click the **Generate Preview** button to view a preview of the files that the Application Builder creates when you build the application. You can use this preview to verify that the Application Builder will create the files you want in the directories you want. You can correct errors before you build the application.
6. Click the **OK** button to close the dialog box and update the project with the build specification settings. The **Tutorial-EXE** build specification appears under **Build Specifications** in the **Project Explorer** window.

Building and Running the Application

1. Save the project. LabVIEW does not save build specifications settings on disk until you save the project.
2. Right-click the **Tutorial-EXE** build specification and select **Build** from the shortcut menu to build the application. A dialog box displays the progress of the build. When the dialog box displays the **Build Complete** message, click the **Done** button.
3. Navigate to the `AB Tutorial\App` directory. The directory should contain the executable file `Tutorial.exe`, the LabVIEW configuration settings file `Tutorial.ini`, and the `data` directory, which contains the support files `datafile.txt` and `lvanlys.dll`.
4. Double-click `Tutorial.exe` to run the application. A file dialog box prompts you to navigate to and select the `datafile.txt` file, located in the `data` directory created with the application.
5. Click the **Acquire Data** button to load data in the buffer. Click the **Graph Data** button to display the data in the graph on the right. Select an analysis to perform in the **Type of Analysis** section and click the **Analyze** button to display statistics on the data.
6. Click the **Stop** button to close the application.

3. Packed Project Libraries

Back to Top

Complete the following steps in each task to configure the build specification settings and build the packed project library. Refer to the *LabVIEW Help* for the most detailed and up-to-date information about building shared libraries. You can access the *LabVIEW Help* by selecting **Help»LabVIEW Help** in LabVIEW.

Configuring the Build Specification

1. From the **Project Explorer** window, right-click **Build Specifications** and select **New»Packed Library** from the shortcut menu to display the **Packed Library Properties** dialog box.
2. Complete the following steps on the **Information** page of the dialog box.
   1. Enter `Tutorial-packedlib` in the **Build specification name** text box. The text box contains the name of the shared library build specification that appears under **Build Specifications** in the **Project Explorer** window.
   2. Enter `Tutorial-packedlib.lvlibp` in the **Target filename** text box. The text box contains the filename on disk for the executable file.
   3. Click the **Browse** button next to the **Destination directory** text box and navigate to the `AB Tutorial` directory. Create a subdirectory named `PackedLib`. Double-click the `PackedLib` directory and click the **Current Folder** button to select it. When you build the shared library, the files appear in this directory.
3. From the **Category** list on the left, click **Source Files** to display the **Source Files** page, which contains settings you can use to specify which files are included and how they are included in the shared library. Complete the following steps on the **Source Files** page of the dialog box.

    1. From the **Project Files** tree, click the **Tutorial Files** folder to expand it. Select `AB Tutorial Library.lvlib`. Click the **Add Item** arrow button next to the **Top-level Library** listbox to add the VI to the listbox.
    2. From the **Project Files** tree, select `AB Tutorial MeanMeadianMode.vi`. Click the **Add Item** arrow button next to the **Always Included** listbox to add the VI to the listbox. Files you add to the **Always Included** listbox are in the packed library, but the packed library calls them dynamically as needed.
    3. Repeat the previous step to add the datafile.txt file to the **Always Included** listbox.
4. From the **Category** list, select the other dialog box pages to explore the settings you can configure in a build specification for a packed library. For this exercise, do not change the settings on these dialog box pages.
5. From the **Category** list, click **Preview** to display the **Preview** page. Click the **Generate Preview** button to view a preview of the files that the Application Builder creates when you build the packed library. You can use this preview to verify that the Application Builder will create the files you want in the directories you want. You can correct errors before you build the packed library.
6. Click the **OK** button to close the dialog box and update the project with the build specification settings. The **Tutorial-packedlib** build specification appears under **Build Specifications** in the **Project Explorer** window.

Building the Packed Library

1. Save the project. LabVIEW does not save build specifications settings on disk until you save the project.
2. Right-click the **Tutorial-packedlib** build specification and select **Build** from the shortcut menu to build the shared library. A dialog box displays the progress of the build. When the dialog box displays that the build is complete, click the **Done** button.
3. Navigate to the `AB Tutorial\PackedLib` directory. The directory should contain the packed library file `Tutorial-packedlib.lvlibp` and the support files `datafile.txt` and `lvanlys.dll`.

4. Shared Libraries

Back to Top

Complete the following steps in each task to configure the shared library build specification and build the shared library. Refer to the *LabVIEW Help* for the most detailed and up-to-date information about building shared libraries. You can access the *LabVIEW Help* by selecting **Help»LabVIEW Help** in LabVIEW.

Configuring the Build Specification

1. From the **Project Explorer** window, right-click **Build Specifications** and select **New»Shared Library** from the shortcut menu to display the Shared Library Properties dialog box.
2. Complete the following steps on the **Information** page of the dialog box.
    1. Enter `Tutorial-sharedlib` in the **Build specification name** text box. The text box contains the name of the shared library build specification that appears under **Build Specifications** in the **Project Explorer** window.
    2. (Windows) Enter `Tutorial-sharedlib.dll` in the **Target filename** text box. The text box contains the filename on disk for the executable file. (Mac OS X) Enter `Tutorial-sharedlib.framework`. (Linux) Enter `Tutorial-sharedlib.so`.
    3. Click the **Browse** button next to the **Destination directory** text box and navigate to the `AB Tutorial` directory. Create a subdirectory named `SharedLib`. Double-click the `SharedLib` directory and click the **Current Folder** button to select it. When you build the shared library, the files appear in this directory.
3. From the **Category** list on the left, click **Source Files** to display the Source Files page, which contains settings you can use to specify which files are included and how they are included in the shared library. Complete the following steps on the dialog box page.
    1. From the **Project Files** tree, click the **Tutorial Files** folder to expand it. Select **AB Tutorial MeanMedianMode.vi**. Click the **Add Item** arrow button next to the **Exported VIs** listbox to add the VI to the listbox and display the **Define VI Prototype** dialog box. LabVIEW exports VIs you add to the **Exported VIs** listbox so you can send and return values to and from the shared library.
    2. Complete the following steps from the **Define VI Prototype** dialog box.
        1. Click the **return value** parameter in the **Parameters** listbox. Verify that the **Function Prototype** description begins with `void`.
        2. Verify that the **Parameters** listbox includes an input named **Array** and outputs named **mean**, **median**, and **mode**.
        3. If the function does not contain these parameters, click the blue **Add Function** (+) button to add the parameters and set the values to input or output.
        4. Click the **OK** button to save the configuration settings and close the **Define VI Prototype** dialog box.

3. From the **Project Files** tree, select **AB Tutorial StndDev.vi**. Click the **Add Item** arrow button next to the **Exported VIs** listbox to add the VI to the listbox and display the **Define VI Prototype** dialog box.
4. From the **Define VI Prototype** dialog box, complete the steps listed above for the `AB Tutorial MeanMedianMode.vi`, but with different parameters. Verify that the **Parameters** listbox includes an input named **Array** and an output named **standardDeviation**.
5. From the **Project Files** tree, select **AB Tutorial Histogram.vi**. Click the **Add Item** arrow button next to the **Exported VIs** listbox to add the VI to the listbox and display the **Define VI Prototype** dialog box.
6. From the **Define VI Prototype** dialog box, complete the steps listed above for the `AB Tutorial MeanMedianMode.vi`, but with different parameters. Verify that the **Parameters** listbox includes an input named **Array** and an output named **HistogramOut**.

4. From the **Category** list, select the other dialog box pages to explore the settings you can configure in a build specification for a shared library. For this exercise, do not change the settings on these dialog box pages.
5. From the **Category** list, click **Preview** to display the **Preview** page. Click the **Generate Preview** button to view a preview of the files that the Application Builder creates when you build the shared library. You can use this preview to verify that the Application Builder will create the files you want in the directories you want. You can correct errors before you build the shared library.
6. Click the **OK** button to close the dialog box and update the project with the build specification settings. The**Tutorial-sharedlib** build specification appears under **Build Specifications** in the **Project Explorer** window.

Building the Shared Library

1. Save the project. LabVIEW does not save build specifications settings on disk until you save the project.
2. (Mac OS X) To build part of a shared library in LabVIEW, Application Builder must use gcc, the standard command-line compiler for Unix and Mac OS X operating systems. Use Xcode, the standard development environment for Apple computers, to install gcc. You can download Xcode for free from the Mac App Store.
3. Right-click the **Tutorial-sharedlib** build specification and select **Build** from the shortcut menu to build the shared library. A dialog box displays the progress of the build. When the dialog box displays the **Build Complete** message, click the **Done** button.
4. Navigate to the `AB Tutorial\SharedLib` directory. The directory should contain the shared library file `Tutorial-sharedlib.dll`, the file `Tutorial-sharedlib.h` that contains parameter prototypes for the shared library functions, the LabVIEW configuration settings file `Tutorial.ini`, and the `data` directory, which contains the support files for the shared library.

You can call the shared library from a number of programming languages, including LabVIEW and C.

5. Installers (Windows)

Back to Top

You can create a single installer that will install the LabVIEW Run-Time Engine, any hardware drivers, and MAX configuration settings necessary for running the application. Complete the following steps in each task to configure the installer build specification and build and run the installer. Refer to the *LabVIEW Help* for the most detailed and up-to-date information about building installers for Windows. You can access the *LabVIEW Help* by selecting **Help»LabVIEW Help** in LabVIEW.

Configuring the Build Specification

1. From the **Project Explorer** window, right-click **Build Specifications** and select **New»Installer** from the shortcut menu to display the **Installer Properties** dialog box.
2. Complete the following steps on the **Product Information** page of the dialog box.
    1. Enter `Tutorial-Installer` in the **Build specification name** text box. The text box contains the name of the installer build specification that appears under **Build Specifications** in the **Project Explorer** window.
    2. Enter `AB Tutorial` in the **Product name** text box. The text box contains the name that displays when the installer runs.
    3. Click the **Browse** button next to the **Installer destination** text box and navigate to the `AB Tutorial` directory. Create a subdirectory named `Installer`. Double-click the `Installer` directory and click the **Current Folder** button to select the directory. When you build the installer, the files appear in this directory.
3. From the **Category** list on the left, click **Source Files** to display the **Source Files** page, which contains settings you can use to specify which files are included and how they are included in the installer.
4. From the **Project View** tree, click the **Tutorial-EXE** build specification to select it. From the**Destination View** tree, click the

TutorialProject folder to select it. Click the **Add Item** arrow button next to the **Destination View** tree to add the application build specification files to the installer.

5. From the **Category** list, click **Shortcuts** to display the **Shortcuts** page, which you can use to create shortcuts to the files that the installer installs and to determine the location of the shortcuts. Complete the following steps on the **Shortcuts** page to create a shortcut on the desktop.
   1. Click the **Add Shortcut** button to display the **Select Target File** dialog box. Double-click the `Tutorial.exe` file to select it.
   2. Change the setting in the **Directory** ring to `[All Users Desktop]` so the shortcut appears on the desktop.
   3. Change the text in the **Name** text box to `Tutorial Application`.
   4. Delete the text in the **Subdirectory** text box so the shortcut appears directly on the desktop and not within a subdirectory.
6. From the **Category** list, click **Additional Installers** to display the **Additional Installers** page, which you can use to include National Instruments product installers that you need in the installer build. The **National Instruments Installers to Include** listbox contains a list of the installers available on the computer.
7. Place a checkmark in the **NI LabVIEW Run-Time Engine** checkbox, if a checkmark is not already present, to include the LabVIEW Run-Time Engine in the installer and all its components. Users who want to run the application must have the LabVIEW Run-Time Engine installed.
8. From the **Category** list, select the other dialog box pages to explore the settings you can configure in a build specification for an installer. For this exercise, do not change the settings on these dialog box pages.
9. Click the **OK** button to close the dialog box and update the project with the build specification settings. The **Tutorial-Installer** build specification appears under **Build Specifications** in the **Project Explorer** window.

Building and Running the Installer

1. Save the project. LabVIEW does not save build specifications settings on disk until you save the project.
2. Right-click the **Tutorial-Installer** build specification and select **Build** from the shortcut menu to build the installer. A dialog box displays the progress of the build. When the dialog box displays the **Installer build complete** message, click the **Done** button.
3. Navigate to the `AB Tutorial\Installer\Volume` directory. The directory should contain subdirectories named `bin`, `license`, and `supportfiles`, and the files `nidist.id`, `Setup.exe`, and `setup.ini`.
4. If you want to run the installer to test the LabVIEW Run-Time Engine installation, copy the files to a computer that does not include LabVIEW and run `Setup.exe`.

6. Source Distributions

Back to Top

Complete the following steps in each task to configure the source distribution build specification and build the source distribution. Refer to the *LabVIEW Help* for the most detailed and up-to-date information about building source distributions. You can access the *LabVIEW Help* by selecting **Help»LabVIEW Help** in LabVIEW.

Configuring the Build Specification

1. From the **Project Explorer** window, right-click **Build Specifications** and select **New»Source Distribution** from the shortcut menu to display the **Source Distribution Properties** dialog box.
2. Complete the following steps on the **Information** page of the dialog box.
   1. Enter `Tutorial-Dist` in the **Build specification name** text box. The text box contains the name of the source distribution build specification that appears under **Build Specifications** in the **Project Explorer** window.
   2. Click the **Browse** button next to the **Destination directory** text box and navigate to the `AB Tutorial` directory. Create a subdirectory named `Distribution`. Double-click the `Distribution` directory and click the **Current Folder** button to select the directory. When you build the source distribution, the files appear in this directory.
3. From the **Category** list on the left, click **Source Files**, which contains settings you can use to specify which files are included and how they are included in the source distribution. Complete the following steps on the dialog box page.
   1. From the **Project Files** tree, click the **Tutorial Files** folder to expand it.
   2. Select all of the items in the folder.
   3. Click the **Add Item** button next to the **Always Included** listbox to add the items to the listbox.
4. From the **Category** list, select the other dialog box pages to explore the settings you can configure in a build specification for a

source distribution. For this exercise, do not change the settings on these dialog box pages.

5. From the **Category** list, click **Preview** to display the **Preview** page. Click the **Generate Preview** button to view a preview of the files that LabVIEW creates when you build the source distribution. You can use this preview to verify that LabVIEW will create the files you want in the directories you want. You can correct errors before you build the source distribution.
6. Click the **OK** button to close the dialog box and update the project with the build specification settings. The**Tutorial-Dist** build specification appears under **Build Specifications** in the **Project Explorer** window.

Building the Source Distribution

1. Save the project. LabVIEW does not save build specifications settings on disk until you save the project.
2. Right-click the **Tutorial-Dist** build specification and select **Build** from the shortcut menu to build the source distribution. A dialog box displays the progress of the build. When the dialog box displays the **Build Complete** message, click the **Done** button.
3. Navigate to the `AB Tutorial\Distribution` directory. The directory should contain the VIs and support files from the project as well as other internal files that LabVIEW determined the source distribution needed to include.

7. Zip Files

Back to Top

Complete the following steps in each task to configure the zip file build specification and build the zip file. Refer to the *LabVIEW Help* for the most up-to-date information about building zip files. You can access the *LabVIEW Help* by selecting **Help»LabVIEW Help** in LabVIEW.

Configuring the Build Specification

1. From the **Project Explorer** window, right-click **Build Specifications** and select **New»Zip File** from the shortcut menu to display the **Zip File Properties** dialog box.
2. Complete the following steps on the **Zip Information** page of the dialog box.
   1. Enter `Tutorial-Zip` in the **Build specification name** text box. The text box contains the name of the zip file build specification that appears under **Build Specifications** in the **Project Explorer** window.
   2. Click the **Browse** button next to the **Destination** text box and navigate to the `AB Tutorial` directory. Create a subdirectory named `Zip`. Double-click the `Zip` directory. Enter `Tutorial.zip` in the file name field and click the **Select** button. When you build the zip file, a file with the name you specified appears in the directory you specified.
3. From the **Category** list on the left, click **Source Files** to display the Source Files page, which contains settings for the files to include in the zip file.
4. From the **Source Files** page, place a checkmark in the **Zip entire project** checkbox so the zip file includes all the VIs and support files in the project. This is a faster option than moving files from the **Project Files** tree to the **Included Items** listbox if you want to include all the project items in the zip file.
5. From the **Category** list, select the Zip File Structure page other dialog box pages to specify the file structure for a zip file. For this exercise, do not change the settings on this page.
6. Click **Preview** to display the Preview page. Click the **Generate Preview** button to view a preview of the compressed file structure that the Application Builder creates when you build the zip file. You can use this preview to verify that the Application Builder will include the files you want from the directories you want. You can correct errors before you build the zip file. Notice that the zip file includes all project items and the project file (`.lvproj`) itself.
7. Click the **OK** button to close the dialog box and update the project with the build specification settings. The**Tutorial-Zip** build specification appears under **Build Specifications** in the **Project Explorer** window.

Building the Zip File

1. Save the project. LabVIEW does not save build specification settings on disk until you save the project.
2. Right-click the **Tutorial-Zip** build specification and select **Build** from the shortcut menu to build the zip file. A dialog box displays the progress of the build. When the dialog box displays the **Build complete** message, click the **Done** button.
3. Navigate to the `AB Tutorial\Zip` directory. The directory should contain the file `Tutorial.zip`.

You can use a standard decompression tool to decompress the zip file and view the VIs and support files it includes.

Back to Top

Customer Reviews

23 Reviews | Submit your review

Why is this part of the best practices series? - 4-apr-2017
By Clendon Gibson, Nabors.
I was looking for advice on build management with LabVIEW and found this. This is clearly a how to article. It does not even go into why you want the various kinds of files.

Completely unusable - 29-mrt-2017
Doesn't explain even the basics of actually deploying a stand alone application. There are 7 steps here, do I need to do all 7 for each application I build?

almost unusable - 8-dec-2016
By Tim Harris, Parker Hannifin.
These instructions are like written by a summer-intern engineering undergrad with no human experience. Explain the names -- why use names like "AB Tutorial" -- or is this a placeholder for a name of users choice? What to do at the end (I was using the "installer" option). Zip a directory and send to user by email? Which directory? Other files (like "alias" that appeared -- can they now be deleted?) Or only send the install file? These are just the operational questions. I leave to the side the absent explanations of numerous small things as well as inaccurate descriptions (e.g. establishing the "destination directory" and so forth).

Desperately needs to be re-written - 11-aug-2016
Not easy to follow and very little in the way of explanation of the method.

Need better more readable help files - 30-nov-2015
This file is impossible to read and follow... I expect better tutorials from NI

Issues with Distributing Applications - 17-okt-2012
What I would like is information on the choices for each case. Which of the options are viable with the exe, source distribution, llb, etc options. What is the effect of the Modify Project File after removing unused members. Does this apply to the destination or the source Project file if you include instrument VIs from an llb? This document just tells how to build the example files. Also, what is required if it is desired to call the VIs in the library from a run time exe program?

Poor quality document; Lacks details; Confuses readers - 25-apr-2012
By Anurag Moghe, AMEADS.
Documentation is not NI's forte. This is a poor quality instruction set. I was looking for a document with a lot of necessary details. The initial part in this document is quite confusing and doesn't yield expected results. I am using LabVIEW 2010 SP1 32-bit and will appreciate a professional quality detailed document from a sincere NI engineer.

 - 13-dec-2011
Simple comment on the Stand Alone Application, what do I need to put on the target computer? It tells me how to run the standalone on the development computer but not on the target computer. That is the whole point of making a standalone. An itemjust before point 3 saying "Copy the folder xxx to the target computer, place in the yyy directory" would be most useful.

 - 12-jul-2011
You have to have version LV2010 to be able to make packed libraries. I have LV2009 and when right clicking build specification there is no packed libraries.

 - 18-apr-2011
I have built a vi and created a dll with it. Then I called the dll in labVIEW,but the output is not right. Specifically, the vi is used to open a device associated with the PC via USB.In the vi,I used the dlls which are provided by the device.After the vi is runed ,I got the information that the device is opened. So there is no problem with the vi.I don't know why the dll created by the vi can't work .

Additionnal Installers do not appear as expected - 27-jul-2010
This tutirial looks quite clear to excecute! However "Additionnal Installers" do not appear as expected so the tutorial cannot be excecuted. Could be an install problem? => Suggestion: missing a step to check if all products are correctly installed and activated

App Builder Tests with LV861 and LV2009SP1 - 24-mei-2010
By Timothy Tower, Anderson Electric Controls.
I followed the above instructions. I used two separate PCs, one with LV861 , and the other with LV2009SP1. 1. (LV861 only): the executable wont run unless one adds error.llb to the project. For some reason, the compiler does not include several key VIs and controls from this llb. Example:

"Not Found dialog.vi." LV2009SP1- works OK without error.lib added. 3. The Source Distrib includes all the VIs but not the lvproj file. One must create a new project for this and save it in the Source folder. (LV861 & LV2009SP1 - same issue). This is tedious, and shouldn't be necessary. 4. The Zip Distrib includes the lvproj but leaves out the dependencies of the top level VI. Why? Hence, a zip file is no use for distributing source code. (LV861 & LV2009SP1 - same issue).

The document has been updated! - 21-aug-2009
Based on the feedback we've received, the above document has been updated with more details and specific information on how to distribute applications with the LabVIEW Application Builder. - National Instruments

This document has been updated - 20-aug-2009
By Nitin T, National Instruments.
Based on the various feedbacks we've received over time, this document has been updated with more details on how to distribute applications using the LabVIEW Application Builder

This is no help! - 11-mrt-2008
I have built a vi and I want to convert it into an executable, but for the life of me there is no step-by-step direction(s) to be found anywhere on HELP or on the NI site. This is unfortunately a common event.

- 23-jan-2008
This document is very complicated and will not give exact result as we want. Why in previous version easy to build EXE but the latest is complicated!!

Issues - 20-jul-2007
By Chuck Streb, GM.
I too am having issues. I cant sem to get a callback function to work at all. More information is needed. Someone needs to read these messages.

- 25-jun-2007
This blows, typical NI documentation. Why is it so hard to create a EXE file?

Building exe's with LV Lib VIs as subVIs - Part II - 24-mei-2007
By Dennis Kessler, Ericsson.
The LabVIEW Runtime Engine 8.2 that I had installed on the target PC using the installer on the distribution CD was not working with these exe's. However, it was able to run an exe on the target PC, when the exe only contains user created sub VIs (no VI Lib files). I had to create an Installer from a Project to re-install the LabVIEW Runtime Engine 8.2. After doing this the more complex exe's now run on the target PC. To all those still searching for a way to build a working exe: Good Luck!

Building exe's with LV Lib VIs as subVIs - Part I - 23-mei-2007
By Dennis Kessler, Ericsson.
I have to agree with Ted Anderson regarding the following problem: "...by specifying the top-level vi, the sub-vis that it calls can NOT be included automatically in the compilation or source distribution?" This is a very serious omission, and the failure to properly document it is inexcusable. Note that the following _is_ successful: Running an exe on the target PC, where the exe only contains user created sub VIs (no VI Lib files). The LabVIEW Runtime Engine 8.2 was installed on the target PC using the installer on the distribution CD. I ran into the problem mentioned by Mr. Anderson when trying to build an exe that calls VI Lib files (Histogram.vi, etc.) as sub VIs. In this case I added a re-named copy of Histogram.vi to my llb and tried to build an exe of the top level VI. (yes, I confirmed that my VI was calling the re-named VI, and that it resided in the llb.) The exe failed to run when launched on the target PC (complaining that the front panel for Histogram2.vi could not be found). Thankfully I read Mr. Andersona's comment and opened all of my sub VIs before creating the Project and building the exe. In Source File Settings I had to select a "Dynamica" for the VI Lib files, and for the one pop-up VI I had. The top level VI is selected as the a 'Top Level VIa' After doing this I was still not able to run the exe on the target PC. The final step was from Chin Lina's email thread in the NI Developer Zone: http://forums.ni.com/ni/board/message?board.id=170&message.id=226748&query.id=22454

Not enough information - 18-apr-2007
By Jeffrey Scharpf, Milwaukee Electric Tool.
I agree with first comment. It does not tell you how to do what you need to do. It tells you that you CAN do it. A step by step example should be provided, along with an explanation of what needs to be exported/configured etc.

What about sub-vis? - 26-mrt-2007
By Ted Anderson, CPP Wind Engineering.
Am I to understand that by specifying the top-level vi, the sub-vis that it calls can NOT be included automatically in the compilation or source distribution? If not, why? Seems like that makes sense, and would obviously be of benefit to the developer. The only way I know of to specify all of the sub-vis is to meticulously/ tediously step through each and every block diagram, find sub-vis, load them, check their properties, go to the project window, find the sub-vi in the directory, and repeat the process over and over and over. Can't believe I'm the first to complain about this. What am I missing?

- 21-feb-2006

Customer Reviews

By Frank Ogletree, Lawrence Berkeley Lab.

This document is OK as an introduction, but more details are needed. The documentation that ships with LV8 has very little detail about using the application builder. The term "Dyanmic VI", used to configure the application builder, is not to be found in the index or searching the documentation. Is this a sub VI? Related to reference nodes? Likewise there are very few details on making an installer.

View more reviews