# Exercise 2-2 Resolve Project Conflicts

## Goal

Use Project Explorer tools to resolve conflicts and manage files in a LabVIEW project.

## Scenario

Conflicts can arise within a LabVIEW project if top-level VIs call incorrect versions of nested code. Applications that are saved in multiple locations as a result of archiving, backup, or division of work, can lead to the use of incorrect code and broken applications.

In this exercise, you examine a LabVIEW project that contains conflicts and use the tools in the Project Explorer to resolve the conflicts and manage the project.

## Design

The project in this exercise contains the following conflicts:

- Two VIs within the project have the same name, `Generate Signal.vi`.
- A VI in the project calls a subVI outside the project, `Log to File.vi`, that has the same name as a VI within the project.

## Implementation

### Part I: Resolving Conflicts

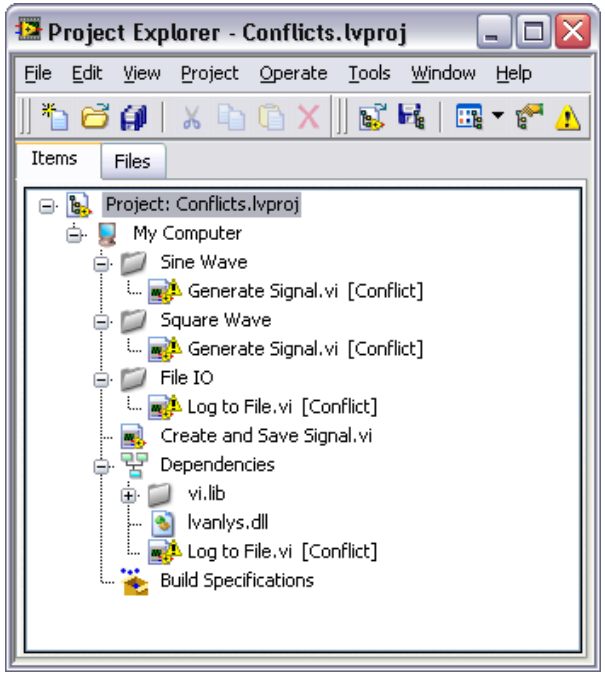
1. Explore a LabVIEW project containing conflicts.

   □ Open `Conflicts.lvproj` in the `<Exercises>\LabVIEW Core 3\Project Explorer Tools` directory.

   □ Expand **Sine Wave**, **Square Wave**, **File IO**, and **Dependencies** in the project tree, as shown in Figure 2-4.

   Notice that LabVIEW has determined that various VIs have conflicts. A conflict is a potential cross-link that occurs when LabVIEW tries to load a VI that has the same qualified name as an item already in the project. When there is a conflict, it is unclear which VI a calling VI should reference.

**Figure 2-4.** LabVIEW Project with Conflicts




- ☐ Double-click **Generate Signal.vi** in the **Sine Wave** virtual folder.
- ☐ Run the VI and observe that this VI generates a sine wave.
- ☐ Close the VI. You do not have to save any changes.
- ☐ Double-click **Generate Signal.vi** in the **Square Wave** virtual folder.
- ☐ Run the VI and observe that this VI generates a square wave.
- ☐ Close the VI.

2. View the file paths of the items in the project tree.

   □ In the **Project Explorer** window, select **Project»Show Item Paths**.

**Tip** The best way to determine if cross-linking exists is to view the full path to the item. Viewing file paths is often the first step in resolving cross-linking conflicts. You can rename or move files as needed, but first you must determine which file is the correct file. Displaying the full path of an item helps you determine the correct file.

   □ Notice that the Generate Signal VI is actually two different VIs with the same name.

   Conflicts might occur if you try to include VIs of the same name because only one VI of a given name can be in memory at a time. If you have a VI with a specific name in memory and you attempt to load another VI that references a subVI of the same name, the VI links to the VI in memory.

3. Determine which VIs in the project call the conflicting VIs.

   □ Right-click **Generate Signal.vi** in the **Sine Wave** virtual folder and select **Find»Callers** from the shortcut menu.

   In the project tree, LabVIEW highlights the Create and Save Signal VI because it calls the Generate Signal VI as a subVI.

   □ Right-click **Generate Signal.vi** in the **Square Wave** virtual folder and select **Find»Callers** from the shortcut menu.

   Even though this Generate Signal VI has no callers in the project, you note that the Generate Signal VI in the `Square Wave` directory performs a different function than the Generate Signal VI in the `Sine Wave` directory. Therefore, renaming one or both files is an appropriate way to resolve the conflict.

4. Manually rename the conflicting files to remove the conflict.

   □ In the **Project Explorer** window, right-click **Generate Signal.vi** in the **Sine Wave** folder, and select **Rename**.

   □ Rename the VI `Sine Wave - Generate Signal.vi` and click **OK**.

   □ Click **Save** when LabVIEW prompts you to save the changes made to the calling VI, `Create and Save Signal.vi`. LabVIEW updates the Create and Save Signal VI to preserve the links to the subVI.

   □ In the **Square Wave** folder, right-click **Generate Signal.vi** and select **Rename**.

   □ Rename the VI `Square Wave - Generate Signal.vi` and click **OK**.

**Tip** You also can rename files from the **Files** page view of the project.

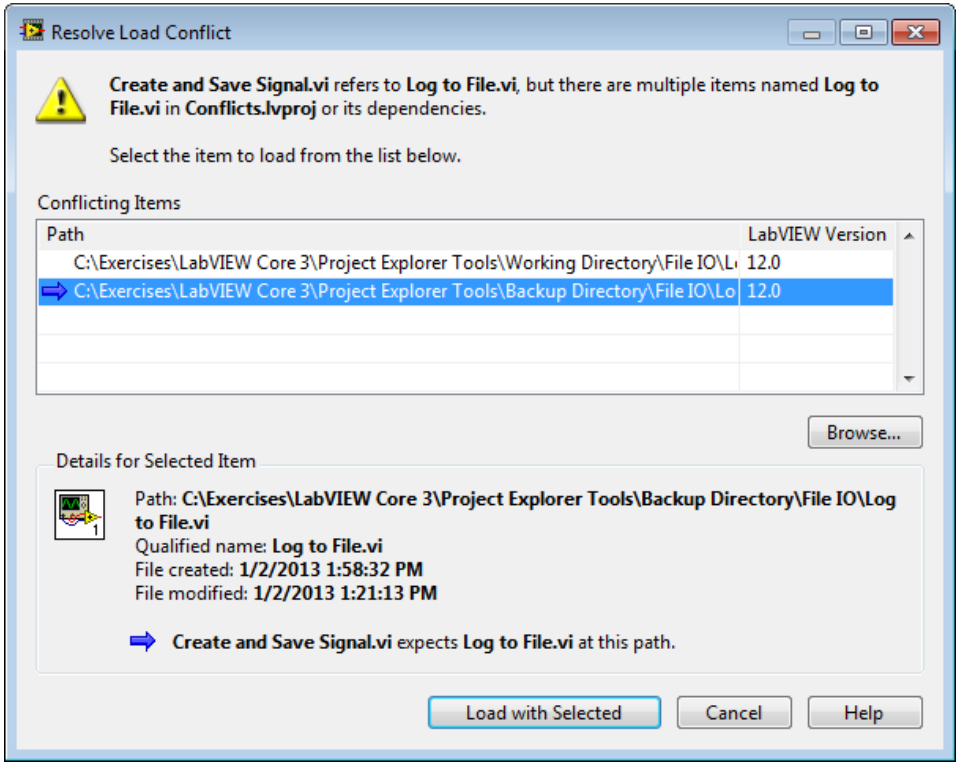
5. Resolve a conflict using the **Resolve Project Conflicts** dialog box.

   ☐ Notice that there is copy of `Log to File.vi` in the **File IO** virtual folder and a copy of `Log to File.vi` in **Dependencies**, which indicates that a file within the project calls the second copy. The file paths of these two VIs are different, as shown in the **Paths** section of the **Project Explorer** window.

   ☐ Right-click each copy of `Log to File.vi` and select **Find»Callers** to determine which file, if any, within the project calls each copy.

   ☐ Double-click **Create and Save Signal.vi** in the LabVIEW Project. The **Resolve Load Conflict** dialog box appears as shown in Figure 2-5 and prompts you to select the subVI you want the caller to reference.

**Figure 2-5.** Resolve Load Conflict Dialog Box



- ☐ In the **Resolve Load Conflict** dialog box, select the Log to File VI in `Working Directory` and click the **Load with Selected** button.
- ☐ Click the **Show Details** button in the **Load Summary Warning** dialog box, which informs you that the Log to File subVI was loaded from a different location.
- ☐ Examine the information display in the **Load and Save Warning List** dialog box and click the **Close** button.
- ☐ Review the items in the **Project Explorer** window. All conflicts in the project should now be resolved.

6. Fix the errors in the Create and Save Signal VI.
   - ☐ The Create and Save Signal VI has unlinked subVIs after resolving the conflicts.
   - ☐ To relink the subVIs, open the block diagram, right-click each subVI, and select **Relink to SubVI** from the shortcut menu.
   - ☐ Save and close the Create and Save Signal VI.
7. Save the project.

## Part II: Exploring Other File Management Tools

1. Use auto-populating folders in the project.
   - ☐ Right-click the **Sine Wave** virtual folder in the project tree and select **Convert to Auto-populating Folder** from the shortcut menu.
   - ☐ Navigate to the `<Exercises>\LabVIEW Core 3\Project Explorer Tools\Working Directory\Sine Wave` directory and click **Select Folder**. Notice in the project tree that the Sine Wave folder icon changes to indicate that the folder is now set to auto-populate.

**Note** In auto-populate mode, the contents of the project folder reflect the hierarchy of the specified folder on disk, as well as any changes that are made outside the development environment.

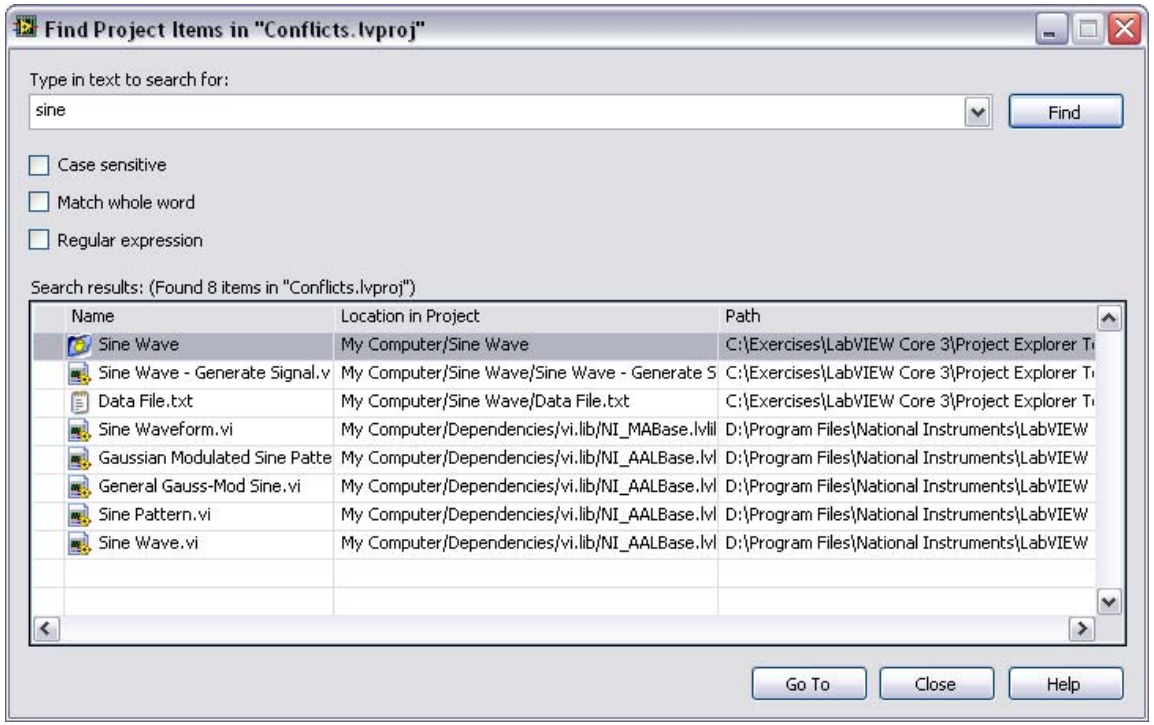
   - ☐ Click the Windows **Start** button and select **All Programs»Accessories»Notepad** to launch Notepad.
   - ☐ In Notepad, select **File»Save** and save the file as `Data File.txt` in the `<Exercises>\LabVIEW Core 3\Project Explorer Tools\Working Directory\Sine Wave` directory.
   - ☐ Close Notepad.
   - ☐ Notice that `Data File.txt` has been added to the **Sine Wave** auto-populating folder on the **Items** page of the **Project Explorer** window.

2. Search for project items.

   □ In the **Project Explorer** window, select **Edit»Find Project Items**.

   □ In the **Find Project Items** dialog box, enter `sine` in the textbox, as shown in Figure 2-6, and click **Find**.

**Figure 2-6.** Find Project Items Dialog Box




   □ Select **Sine Wave - Generate Signal.vi** and click the **Go To** button. **Sine Wave - Generate Signal.vi** is now highlighted in the **Project Explorer** window.

3. Save and close the project.

## End of Exercise 2-2