Calling a Dynamic Link Library (DLL) from LabVIEW

Updated Dec 4, 2019

Reported In ⓘ

Software

- LabVIEW

Programming Language

- C
- C++
- C# .NET

Issue Details

I have code/API written in C/C++ and I want to be able to call it in LabVIEW. Once I have made a Dynamically Linked Library (DLL) with my C code, how do I call the DLL from LabVIEW?

Solution

To call a DLL, you first must identify if the library is a C/C++ DLL or a Microsoft .NET Assembly/.NET DLL.

To determine what type of library you have, you can check the Portable Executable (PE) header for the DLL files.

DLL is a C/C++ DLL:

1. Find the header file (*.h) or function prototype definitions, if you do not have a header file skip to Step 4.
2. Identify the function(s) you wish to call. Notice whether the parameters for the function contain primitive data type parameters (such as int, char, double, etc.) or whether they contain complex data types (such as a struct, array, or vector).
3. If the function does not contain any complex data types and you have a header file (*.h), you can use the Import Shared Library Wizard to create a LabVIEW library containing the functions in the header file. You can find this option in **Tools»Import»Shared Library (.dll)...** Continue with the Wizard.
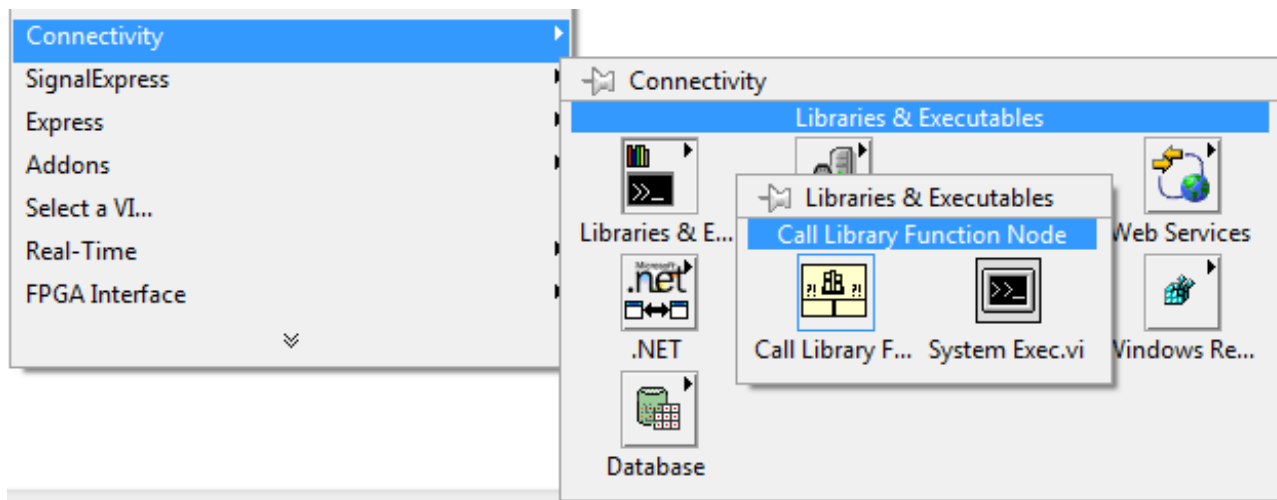
|		
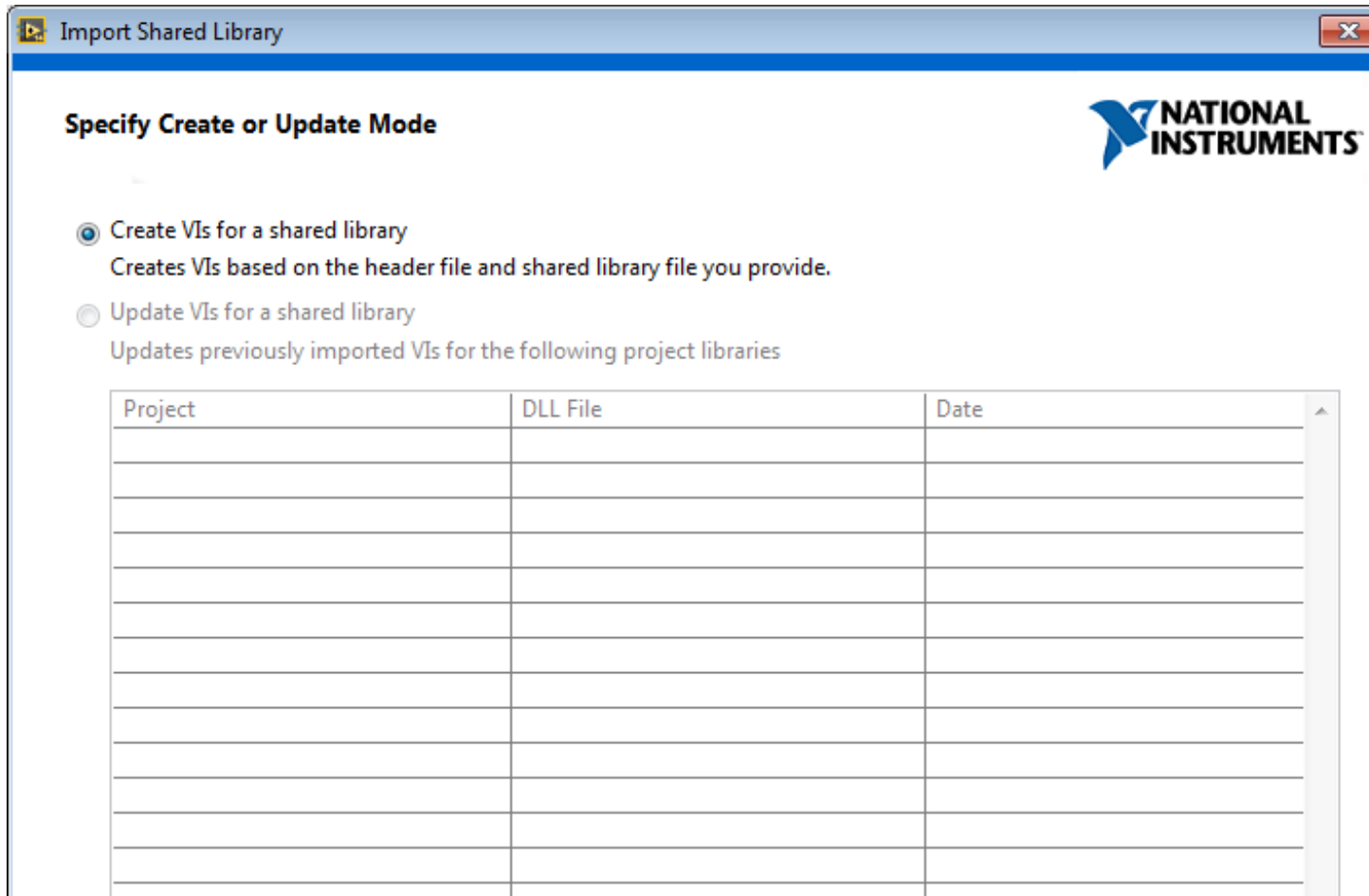
[Back]   [Next]   [Cancel]   [Help]

4. Without a header file, you will need to use the Call Library Function Node along with proper documentation of the .DLL. The picture below shows the location of the Call Library Function Node in the pallet:
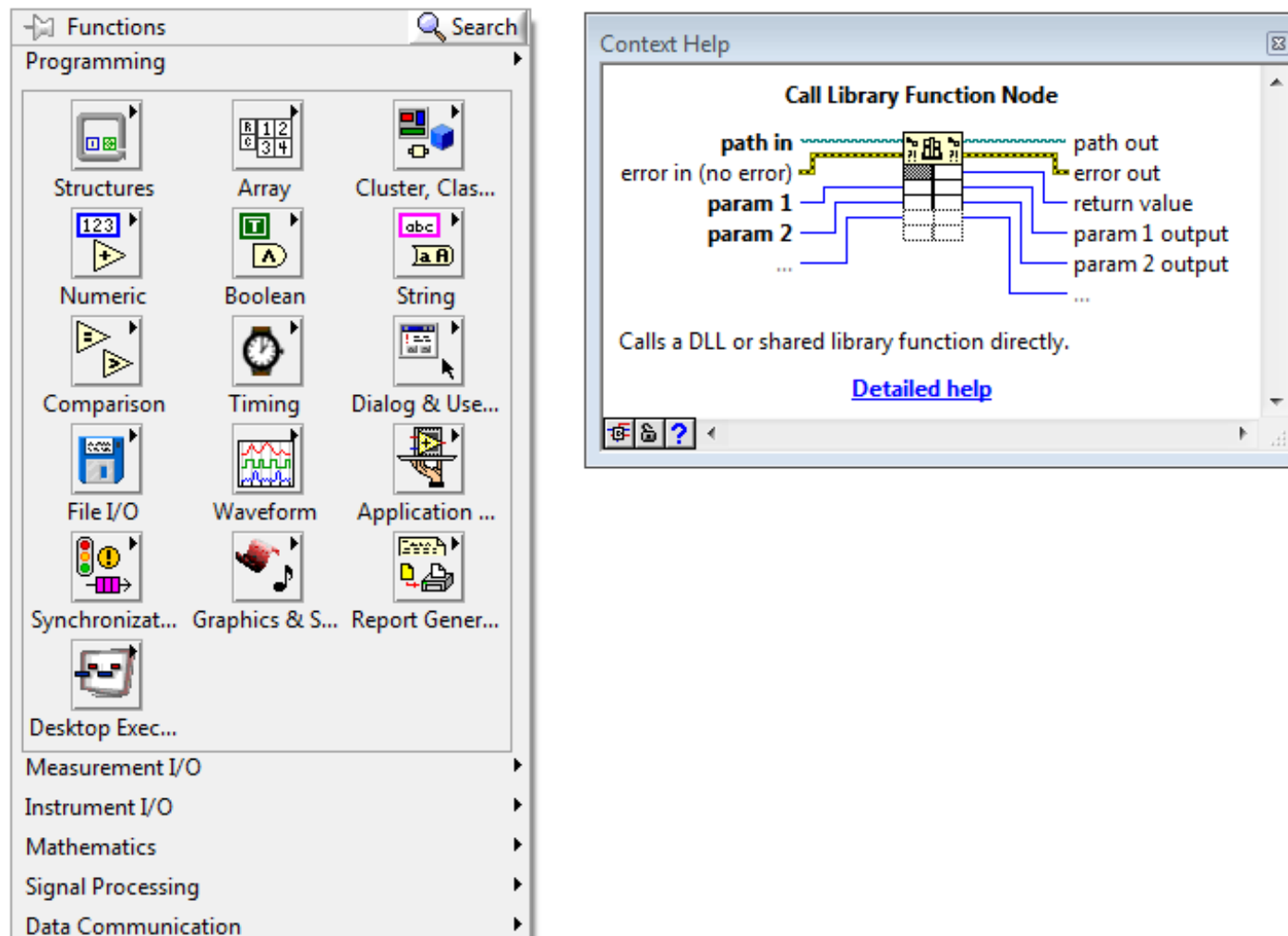
5. Double-click the Call Library Function Node to configure the node. On the **Function** tab of the configuration window, enter the path to the DLL and select the function you wish to call.
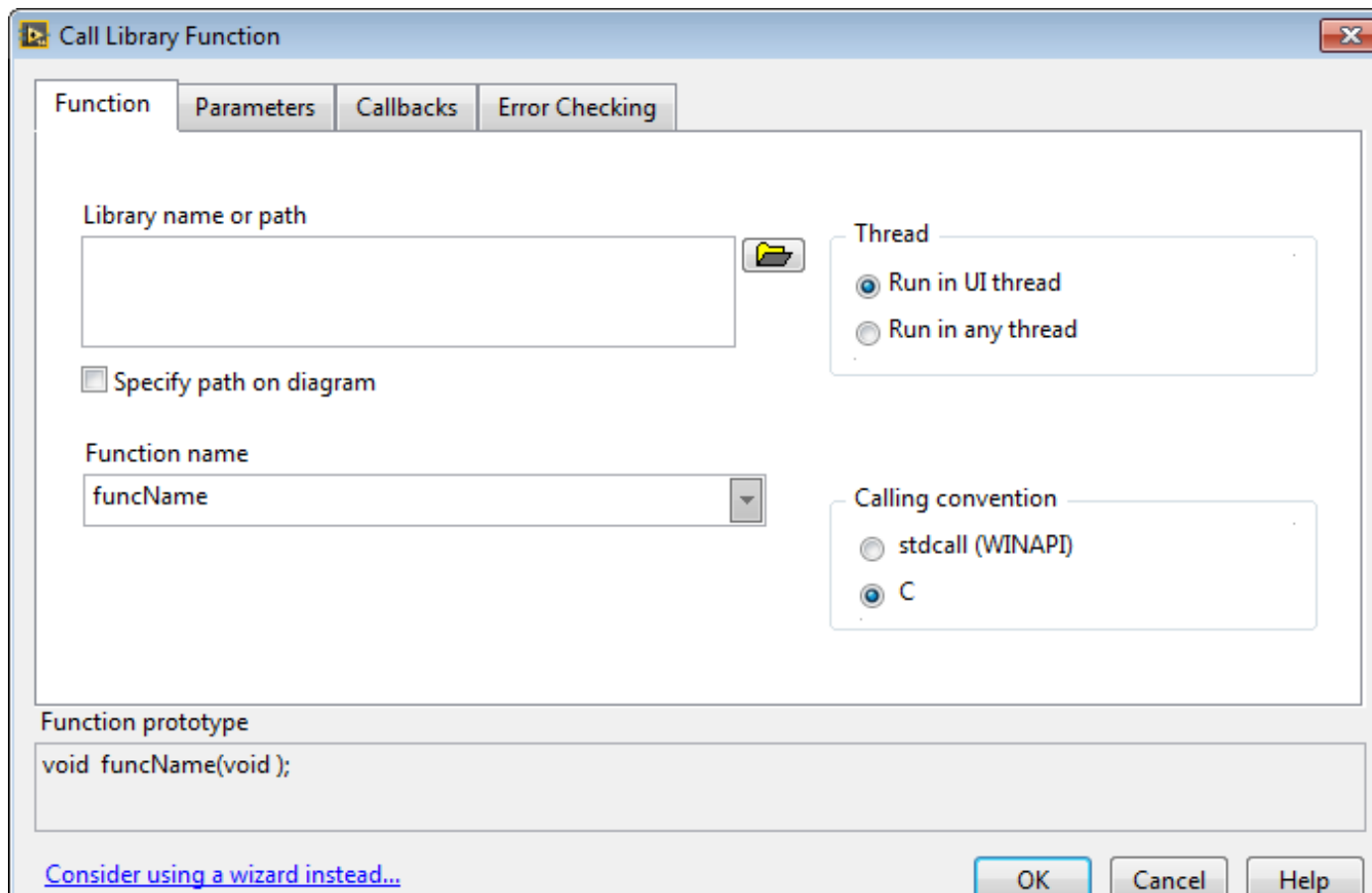
6. If the function(s) you wish to call contains complex data types, you'll need to manually define the function prototype within the configuration dialog in the Call Library Function Node (if the DLL has been compiled with a Type Library, the parameter data and return types will be defined automatically). Navigate to the **Parameters** tab in the configuration window. Here you will add parameters and modify the return type until the function prototype at the bottom of the window matches the DLL's function definition.
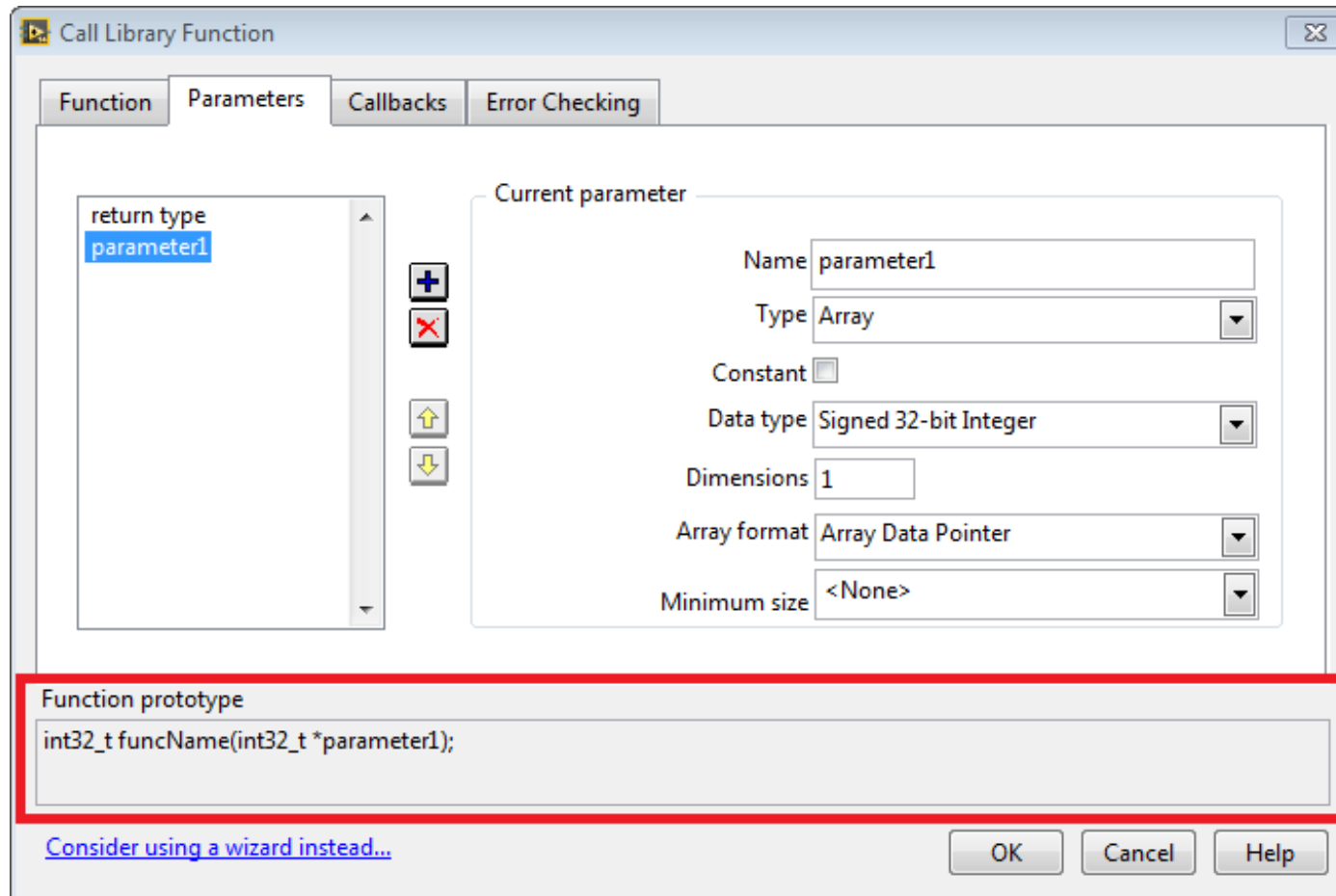

DLL is a .NET assembly:

1. In order to use a .NET assembly in LabVIEW, simply use the .NET palette **(Connectivity».NET)** to find all of the functions available.
2. First use a Constructor Node in order to instantiate a class within the .NET assembly. This can be done by double clicking the

Constructor Node to bring up the Select .NET Constructor dialog box.
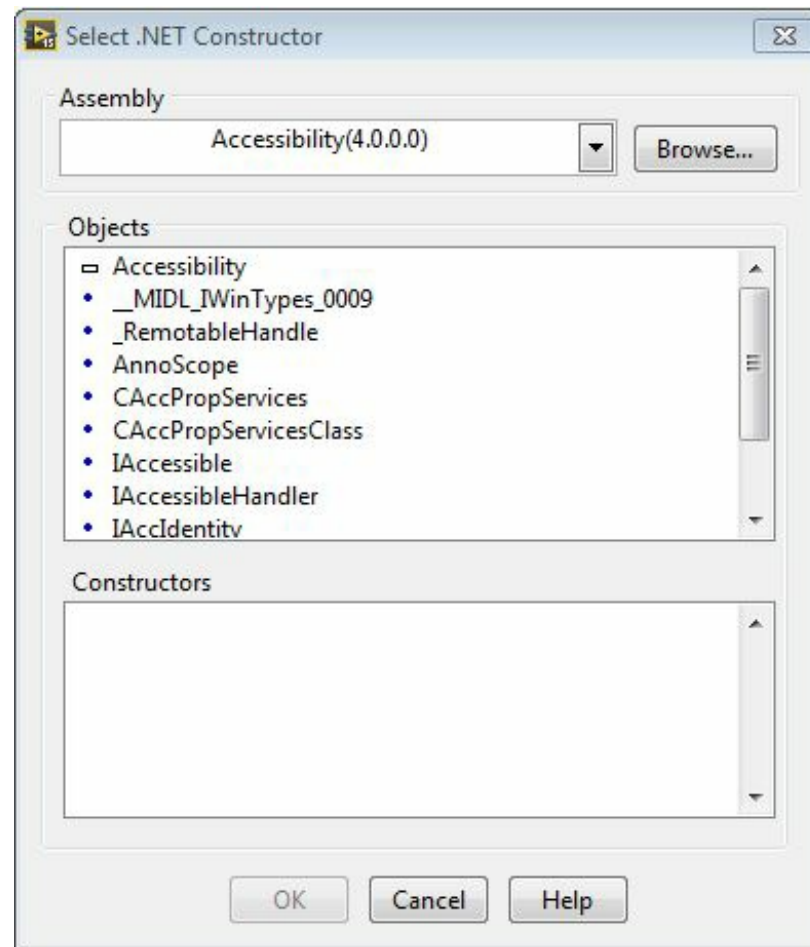
3. Use Property and Invoke Nodes to access properties and methods of the class by simply wiring in the class reference from the Constructor Node to the property or Invoke Node. This is shown in the below VI Snippet.
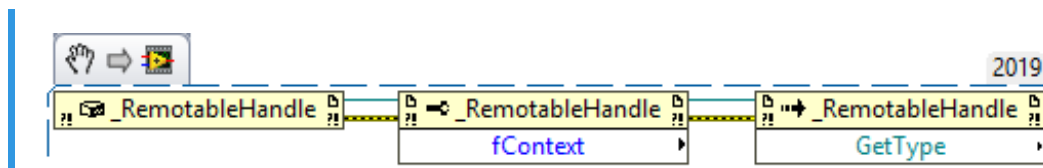
Note: This image is a LabVIEW snippet, which includes LabVIEW code that you can reuse in your project. To use a snippet, right-click the image, save it to your computer, and drag the file onto your LabVIEW diagram.

Additional Information

A .NET DLL is also called a .NET assembly, which are useful in .NET programming. A .NET DLL uses the Common Language Runtime (CLR) and the .NET Framework to manage the functions within the DLL.

Note that if your complex data type is a struct with more than primitive data types (int, double, char), you can create a wrapper DLL from a C-based language to simplify function calls.

C/C++ constructs do not always directly correlate to LabVIEW data types, and a wrapper DLL can appropriately extract the contents of the complex structure and translate them to primitive terms. For instance, a DLL you are using utilises a struct in C which contains a char * (a string or character array). Your wrapper DLL can contain a wrapper function with a char * parameter that places the char * into a struct, and in turn calls the original DLL. In LabVIEW, you can instead call the wrapper DLL functions. Alternatively, if you have the DLL source code, you can directly modify the DLL so that it takes in a char * instead of a struct.

The LabVIEW example finder has complete and functional examples of how to properly call external code in LabVIEW. See **Help >> Find Examples >> Communicating with External Applications**

Related Links

- Call Library Function Returns the Wrong Function Prototype and Function Parameters for DLLs

- Wrapper VIs for C/C DLL functions using the Import Shared Library WizardCreating

- External Link: Assemblies in the Common Language Runtime

- Calling External APIs

Other Support Options

Ask the NI Community

Collaborate with other users in our discussion forums

- Search the NI Community for a solution

Request Support from an Engineer

A valid service agreement may be required, and support options vary by country.

- Can LabVIEW C? - Example 3: Using the Right Tools with LabVIEW
- Use an Array Pointer Returned by a C DLL in LabVIEW
- What is a Type Library and How is it Used by NI TestStand?

WAS THIS ARTICLE HELPFUL?

Helpful

Not Helpful