

How to use VEE Programs with .NET

Test and Measurement Opportunities within the Microsoft Visual Studio .NET Framework

A new programming duo brings flexibility, ease-of-use and enhanced productivity to test and measurement engineers.

Together the Agilent VEE and the Agilent T&M Programmiers Toolkit for Visual Studio .NET comprise parallel options for programming test and measurement instruments. VEE is an easy-to-learn, measurement-smart PC graphical programming environment designed for individual engineers and scientists at an unprecedented price/performance point. It can connect with up to four instruments for design characterization, design verification and data acquisition.

Historically, users have chosen graphical programming languages for one or more of the these reasons:

- Productivity.
- Modeling metaphor.
- Company standard.

The T&M Toolkit provides a similar level of productivity as VEE, but is based in a more open, industry-standard environment. Since most users of VEE still need a textual programming language, the two complement each other and enable users to program in their language

of choice. Our goal with the T&M Toolkit was to develop a textual programming metaphor that is just as productive and easy to model as a graphical programming metaphor.

The T&M Toolkit provides a test and measurement layer integrated into Visual Studio .NET, and .NET enables a user to program in any of more than 20 languages. Users acquire productivity, flexibility and code re-use all in a single, open environment. With the inherent productivity benefits of Visual Studio, .NET and the T&M Toolkit; Agilent removes the plumbing and overhead issues. So users can focus on their engineering contributions – rather than not being tied up in wiring the pieces together.

VEE will continue to be enhanced and supported as evidenced by the recent 6.1 release. Other follow-on, VEE releases are planned that will bind VEE even closer to the Visual Studio .NET environment. Over time, Agilent plans to support a use-model in Visual Studio .NET that provides an additional migration path for VEE users.

For those who have developed VEE code - or relate better to the modeling metaphor of graphical programming, VEE is clearly the best

choice. But Visual Studio .NET is a better choice in these cases: If an engineer is more familiar with Visual Basic, Visual C++, or Visual C# - is looking for a high degree of program code re-use - wants the flexibility of many different programming languages built on the same foundation - wants to adopt XML - or programs for the Web.

The T&M Toolkit makes it easy for the engineer to wrap code developed in VEE and callable from the Visual Studio .NET environment. While not a completely integrated environment, the two cooperate well. The rest of this paper details how to make your VEE application usable within the .NET Framework.

Preparing VEE Programs for Use with .NET

The T&M Toolkit can generate a VEE wrapper that lets you access the UserFunctions in any (6.03 or greater) VEE Pro program, without requiring you to modify the VEE program. However, if you are willing to revisit your legacy VEE programs, here are some steps you can take to enhance the interaction between your VEE code and your Visual Studio .NET applications:

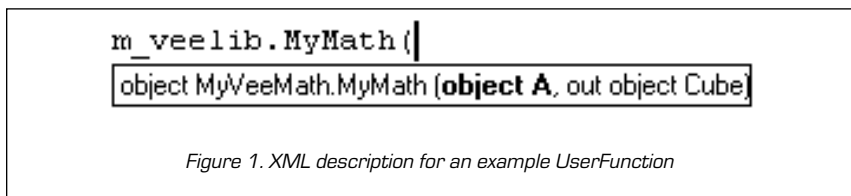
- The Callable VEE Server cannot access the "Main" portion of a VEE program. Only UserFunc-

tions and things embedded within UserFunctions are accessible when calling VEE from a .NET application. Structure your VEE program so that all the features you want to access are built as UserFunctions.

- When naming UserFunctions and their parameters, follow the .NET naming rules. For instance, the first character of a name should be a letter, and a name should contain only letters, numbers, and underscores. It is also helpful to use meaningful names. The VEE Wrapper Wizard automatically alters names that do not conform to .NET naming rules. You are not required to change non-conforming names, but you might prefer your own alterations to the automatic alterations.
- If you know the expected data type for your input parameters, make those constraints part of your program. In VEE, double-click the input terminal to open the "Input Terminal Information" dialog. Set the "Required Type" and "Required Shape" to the expected values. This lets your .NET application see the actual data type, for example, integer or complex. Without this constraint, .NET applications see the terminals as the generic type "object".
- You have the option of creating some of the IntelliSense for your VEE UserFunctions. This is not required, but you might find it handy. This feature is explained in the following paragraphs.

IntelliSense is a Microsoft feature, which displays helpful information in pop-up boxes as you type or hover in the Visual Studio code editor. The following graphic (**Figure 1**) depicts a screen detail that was captured while typing a MyVeeMath example program. The box below the cursor is the IntelliSense Help that appeared when the opening parenthesis was typed. The box shows some basic information about the parameters.

This information box can be enhanced if you choose to include



IntelliSense code in the Description of your VEE UserFunctions. The "Description" dialog is available from the Object Menu of your VEE UserFunction. If you write the Description using the XML elements <summary> and <param>, the VEE Wrapper Wizard creates an XML file that includes your customized IntelliSense information. The graphic in **Figure 1** is an XML description for this example UserFunction.

The format of an XML description can be deduced from this example. The <summary> element must come first. It is the description for the entire UserFunction. Each parameter should have a corresponding <param> element. The exact parameter name is included in the tag using a "name=" attribute. As with all XML files, you must explicitly close every element you open.

program and re-save the program. Run the VEE Wrapper Wizard using the new VEE program. It creates an XML file with the same basename as the wrapper (DLL) file. This XML file augments the IntelliSense Help, as shown in the final graphic **Figure 3**.

There are several things to keep in mind if you choose the XML technique for your descriptions.

- If you start with XML, you must stay with XML. In other words, when the VEE Wrapper Wizard sees <summary> at the beginning of a VEE Description, it treats all the text in that Description as XML. This implies that all text must be included in valid XML tags. For example, if your Description includes more than just the <summary> and <param> elements, the additional text should be in a <remarks> element. Also, special characters must be typed as entities.

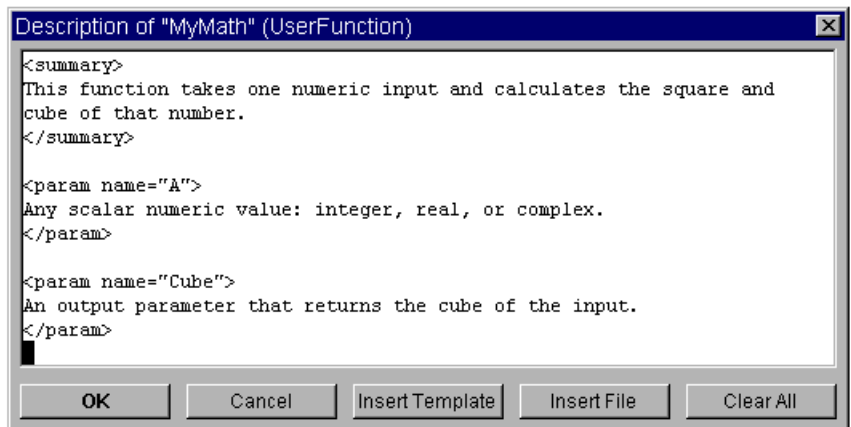
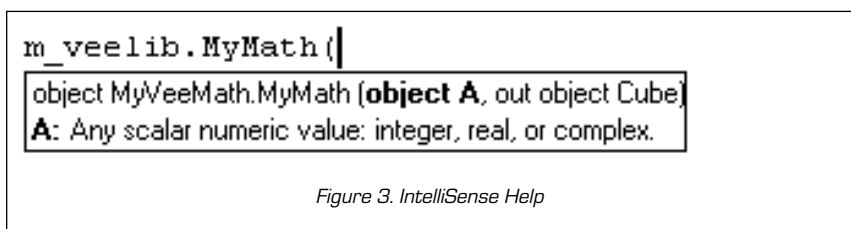


Figure 2. Component interaction between .NET Application and VEE Pro

For enhanced IntelliSense, use this format to create descriptions for all the UserFunctions in your VEE pro-

gram. For example, the character "&" is typed "&#amp;" and the character "<" is typed "<".



- If the wrapper DLL file is copied or moved, copy the XML file to the same folder. Visual Studio needs to find both the DLL and XML files in the same folder when it prepares the IntelliSense Help.

How Do I Wrap My VEE Program and Call It's User Functions?

An Agilent VeeWrapper class enables you to access your Agilent VEE Pro UserFunctions from any .NET CLS-compliant language, such as Microsoft Visual Basic .NET or C#. The VEE Interop classes provide mechanisms for loading and unloading libraries of existing UserFunctions, calling those functions, inspecting the input and output of functions, and translating data types between VEE's special containers and standard .NET types.

Note: To inter-operate properly with .NET languages, your callable VEE programs must use a VEE engine that is version 6.03 or greater.

Preparing a .NET program to communicate with Agilent VEE Pro involves four basic steps. Those steps are detailed in the following discussion, using Visual Basic .NET:

Step 1 - Create a Project Using the Agilent T&M Programmer's Toolkit

Select an "Agilent T&M Toolkit" project type when creating a new project. When the New Project Wizard asks you to select namespaces, be sure to include Agilent.TMToolkit.VeeInterop (you probably do not need Agilent.TMToolkit.VeeInterop.Desig). The New Project wizard adds all the utilities and references you need to communicate with VEE Pro.

Step 2 - Generate a .NET Wrapper Class for Your VEE Program

A VEE wrapper is the primary object used to communicate with your VEE program from .NET. A unique VEE wrapper is built for each VEE program (.vee file) you want to access. To build a VEE wrapper, use

the VEE Wrapper Wizard as follows:

1. Click the VEE icon on the T&M Toolkit toolbar to launch the VEE Wrapper Wizard, or select "VEE Wrapper Wizard" from the T&M Toolkit menu.
2. Enter the path name to the VEE User Function Library (your .vee program file).
3. The wizard parses the VEE program to find all the UserFunctions. Select the UserFunctions you want to access via this Wrapper Class.
4. You can enter a name for the Wrapper's assembly and class. Enter a name for the class you want to build and the path to your .vee file.



Step 3 - Create an Instance of Your VeeWrapper Class

The exact method used to create this object is dependent on the programming language. However, the same general procedure is used in most applications. The following is a summary of that general procedure.

1. The VEE Wrapper Wizard created a .NET assembly and added the necessary references to your project. You supplied a class name - and optionally a namespace name - when you ran the Wizard.
2. Declare a variable of the type specified for your VeeWrapper class.
3. Create an instance of this class. It is most efficient to create this object with a relatively high scope and long lifetime. This avoids the overhead of creating and disposing VeeWrapper objects every time you want to call a function.

Here are some short code segments that show the syntax for declaring and creating a VeeWrapper object:

```
[Visual Basic]
' In the class variables
Dim MyLib As MyVeeMath

' In the Form_Load Sub
MyLib = New MyVeeMath()

[C#]
// in the class variables
private MyVeeMath m_veelib;

// in the form constructor
m_veelib = new MyVeeMath();
```

Step 4 - Call a User Function from the VeeWrapper

A UserFunction is a specific kind of VEE object found in VEE's Device menu. A UserFunction must already exist before the VeeWrapper can access it. In other words, you cannot create a VEE UserFunction from Visual Studio. You must create the VEE UserFunction from VEE.

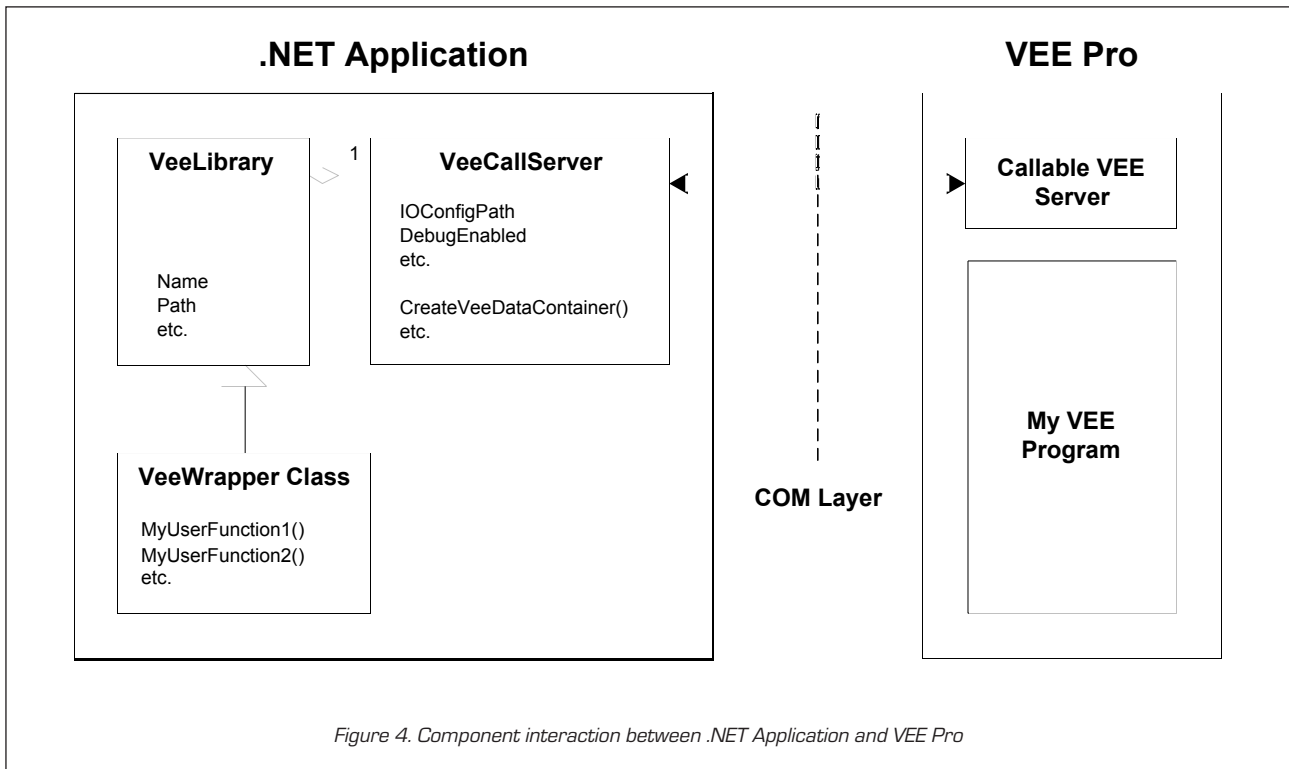
Note: VEE has several objects with similar names. There are "Built-in Functions," "User Functions," and "User Objects." When developing VEE code for a VEE Wrapper Class, be sure to choose "User Function."

The VeeWrapper object provides a method for each UserFunction in your VEE program. When you type the name of your VeeWrapper in the code editor, then type a period, IntelliSense shows you the list of VEE UserFunctions available.

Overview of the Callable VEE Feature

The diagram in **Figure 4** is a simplified view of the essential components and the way they interact when a .NET application communicates with a VEE Pro program. Refer to this figure as you read the explanation that follows it.

VEE Pro provides an ActiveX Automation Server that makes VEE UserFunctions available for use by other applications. Those UserFunctions are stored in a .vee file. The Agilent T&M Toolkit provides classes, methods and properties that enable communication with the Automation Server in VEE. These VEE communication tools are availa-



ble when you add a VeeWrapper class to your .NET application. The VeeWrapper class is created when you run the VEE Wrapper Wizard.

Several classes are involved in this communication with VEE Pro. The most significant classes are shown in **Figure 4**. Here is an overview of their function:

- The VeeLibrary object knows the name and location of your .vee program file. It comprises a collection of all the UserFuctions available in that program. It creates and owns an instance of the VeeCallServer to handle the actual communication with VEE Pro.
- The VeeCallServer object orchestrates all the details of communicating with the VEE application. It knows how to pass data to and from the VEE program, it can control VEE's configuration (such as the window geometry) and provides tools that allow the .NET language to working with high-level VEE data types (such as Spectrum).
- The VeeWrapper is the principal object you use to write code that communicates with VEE Pro. For programming convenience, the VeeWrapper presents each VEE UserFunction as a

separate method, complete with IntelliSense help. In addition, because the VeeWrapper class inherits from the VeeLibrary class, it also provides a way to access all the public members of VeeLibrary and VeeCallServer.

Summary

By following some simple design rules for your VEE application, you can easily extend your VEE environment to the Microsoft .NET Framework. By adding the Agilent T&M Programmers Toolkit to the .NET Framework, you'll find you can create solutions not possible through VEE alone, while leveraging all of the advantages that .NET offers.

About the author

Bill Brecht is a Technical Marketing Engineer with Agilent Technologies. He has worked with Agilent in system design automation for exploration of computer architectures and is currently working with the Measurement and Analysis, Software and Technology Operation.