Agilent VEE Extensible VEE Object (EVO) Developer's Guideline

Application Note

Overview

Agilent VEE functionality has been extended through the use of Extensible VEE Objects (EVOs). Each EVO can have its own graphical user interface (GUI) control and execution behavior. EVOs are very similar to generic VEE objects where there are pins for input, output, control, sequence input, and sequence output which cannot be found on imported .NET controls. Some vTools components are created using EVOs. vTools are complimentary toolboxes that you can use with Agilent VEE 9.3 and help in VEE program development.

EVO is a beta release in Agilent VEE version 9.3, allowing users to write their EVOs for integration into Agilent VEE. EVOs are developed using the Microsoft® .NET add-in platform and can be written using any Microsoft .NET Framework language such as Visual C#® and Visual Basic® .NET.

Table of Contents

Overview 1
How EVO Works in VEE 2
    For end users to use your EVO in Agilent VEE IDE 3
    For end users to run programs containing EVO 3
    EVO terminals 3
System Requirements 4
Creating an EVO 5
Deployment 7
    EVO assembly path 7
Recommended Practices 9
    EVO icon 9
    EVO assemblies path location 9
    Avoid duplicate terminal names 10
    Output terminal name vs error output terminal 10
    Dynamic terminal 10
How Do I Debug EVO Code? 11
Some Notes on the Expression Graph Example 12
Appendix A: Agilent.Vee.Extensibility.IExtensibleVeeObject Interface 13
Appendix B: EVO Configuration File (.evo) 14
Agilent Advantage Services .... Back cover
Contact Agilent .... Back cover

Agilent Technologies

How EVO Works in VEE

In order to write an EVO, two things are needed:

1. A .NET class needs to implement the Agilent.Vee.Extensibility.IExtensibleVeeObject interface. This interface defines the contract needed for VEE to host the EVO. Appendix A lists the interface details and descriptions.
2. An EVO configuration file with extension .evo is needed to describe the EVO. Appendix B lists the layout, contents and descriptions of the EVO configuration file. One EVO configuration file is needed for each EVO.

The EVO configuration file must reside in either location below:

Table 1. EVO configuration file locations

Operating system	Directory	User visibility in **Device > Extensible Vee Objects** menu
Windows XP	C:\Documents and Settings\{User}\My Documents\Agilent\Agilent VEE 9.3\Addins	*Only user {User}*
Windows Vista or Windows 7	C:\Users\{User}\Documents\Agilent\Agilent VEE 9.3\Addins	
Windows XP	C:\Documents and Settings\All Users\Application Data\Agilent\Agilent VEE 9.3\Addins	All users
Windows Vista or Windows 7	C:\ProgramData\Agilent\Agilent VEE 9.3\Addins	

If the directory does not exist in your system, you may need to create it manually and then place the EVO configuration file in the directory. If an EVO configuration file exists in both locations above, it will have two entries in the **Device > Extensible Vee Objects** menu.

When VEE starts up, it browses the EVO configuration file locations, shown in Table 1, for EVO configuration files. If the EVO configuration file format and version match, VEE displays the EVO in the **Device > Extensible Vee Objects** menu. However, in **Device > Extensible Vee Objects** menu, it appears as a flat list and there is no method to organize its appearance in a hierarchical format.

When the EVO is added to the VEE workspace, VEE loads the EVO. The EVO might take longer to load the first time, due to .NET Framework Just-In-Time (JIT) and runtime constructions. It will take less time for subsequent loads. When the program is saved, the EVO is saved as a .NET component, where its .NET assembly's full name and full directory path are saved in the VEE program. In other words, the EVO's full absolute path is saved inside the VEE program. Due to this behavior, there are two scenarios you need to consider when sharing or deploying your EVO:

- When the end user is to use your EVO in Agilent VEE Integrated Development Environment (IDE)
- When the end user is to run a program that contains your EVO

For end users to use your EVO in Agilent VEE IDE

As explained above, in the **Device > Extensible VEE Objects** menu, EVOs appear in a flat list. However, sometimes it is more appropriate to show EVOs in a hierarchical representation, which can be achieved in other VEE menus. For example, in vTools, EVOs are grouped in categories for a better user experience. To create a hierarchical view:

- Add the EVOs into Main.
- Add the necessary VEE objects e.g. notepad and text constant.
- Save them into a VEE program.
- Use the VEE customize menu (.mnu file or .xmnu file) to create a hierarchical menu presentation. Please refer to the VEE help file **Contents > To Customize the VEE Menus**.

In this scenario, the EVO assembly, the VEE program file and customize menu file (.mnu or .xmnu) are needed. The EVO assembly needs to exist in target machine with the same absolute path that is saved inside the VEE program. The EVO configuration file (.evo) can be excluded because the EVO .NET assembly's full absolute path is saved in the VEE program.

If you are planning on creating an installation package for your EVO package, the Deployment section of this document provides information on the steps and details for a seamless end user experience when using EVO and running programs containing EVO.

For end users to run programs containing EVO

If you are using EVOs when writing a VEE program and need to deploy the final program only to other users or other PCs, the EVO configuration file (.evo) is not needed. This is because the EVO assembly's absolute path is saved in the VEE program. In this scenario, in addition to the VEE program, the EVO assembly needs to exist in target machine with the same absolute path that is saved inside the VEE program. However, VEE has enhanced this mechanism and the details are discussed in the Deployment section of this document.

EVO terminals

Each EVO uses a fixed set of data terminals which is defined in DataInputs, DataOutputs, and ControlInputs. All data input terminals must be added and connected, or else there will be compilation errors.

Control input terminals can be left unconnected, thus some EVO developers use them as optional input terminals. When the user adds EVO control input, VEE invokes ControlEvoTerminalAdded. Then the EVO developer can handle the required logic, for example disable some control. Similarly, the removed method is available via ControlEvoTerminalRemoved. Similar methods are available for data input terminals, i.e. InputEvoTerminalAdded and InputEvoTerminalRemoved. However, control input terminals have a different data propagation behavior. For details, refer to VEE help file **Contents > Tell Me About... > Propagation > Handling Propagation Problems > Data Propagation on Control Pins**.

Similar to other VEE objects, data output terminals can be left unconnected.

EVO terminals *(continued)*

For some EVO functionality there is a need to be able to update the EVO terminal list during VEE program development. This can be done by using the TerminalChanged event in the IExtensibleVeeObject interface. We called this a dynamic terminal in EVO. With this feature, we recommend avoiding the use of EVO control input terminals. In most cases, using the dynamic terminal is sufficient.

EVO terminal behavior is slightly different from other VEE object terminal behavior. For example, in the Formula object the user is allowed to add any number of input terminals as needed. This is different in EVO since its terminals are fixed set terminals written in .NET. If there is a need to allow users to add any number of terminals with a customized name, it’s recommended that the dynamic terminal feature be used. Also note that the EVO terminal name cannot be edited. Avoid duplicated terminal Names. More details can be found in Avoid Duplicate Terminal Names.

System Requirements

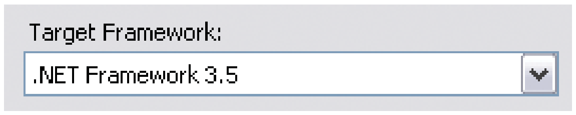
Agilent VEE version 9.3 and above
Microsoft Visual Studio® 2008 and above

Creating an EVO

In this section we will be describing the steps to create an EVO using a C# expression graph example. This example can be downloaded from **http://www.agilent.com/find/veesamples**

1. Create a new Microsoft Visual Studio C# control library. This will be either a WPF User Control Library or a Windows Forms Control Library depending on the EVO developer.

2. At Project Properties, change the Target Framework to .NET Framework 3.5, 3.0 or 2.0.



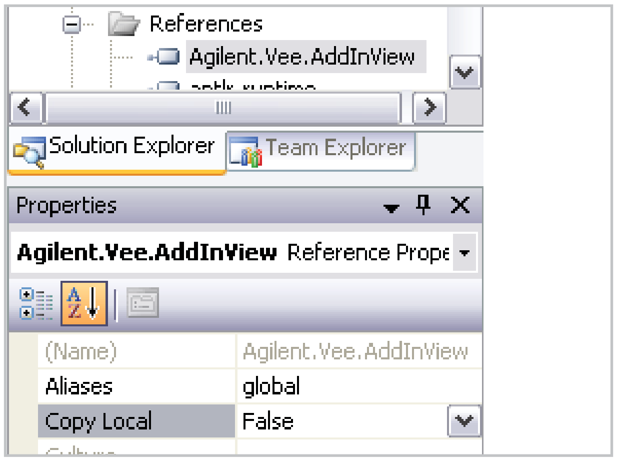
Target Framework:
.NET Framework 3.5

3. Add a reference to **{VEE Installed Directory}\ AddInViews\Agilent.Vee.AddInView.dll**. Typically it is **C:\Program Files\Agilent\VEE Pro 9.3\AddInViews\Agilent.Vee.AddInView.dll.**

   In 64-bit OS, it could be **C:\Program Files (x86)\Agilent\VEE Pro 9.3\AddInViews\Agilent.Vee.AddInView.dll.**

   When compiling the example program, if there is compilation error on missing Agilent.Vee.AddInView, re-add the reference.

4. In Solution Explorer's References, select the DLL just added and change the **Copy Local** to **False**. If this change is not done, it could yield a compilation error later.

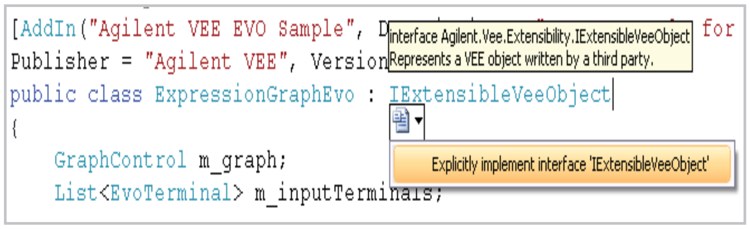


References
Agilent.Vee.AddInView

Properties
Agilent.Vee.AddInView Reference Prope

(Name)	Agilent.Vee.AddInView
Aliases	global
Copy Local	False

5. EVO class implements the **Agilent.Vee.Extensibility.IExtensibleVeeObject interface**. This can be done quickly using the Visual Studio-provided implement interface options.



```
[AddIn("Agilent VEE EVO Sample", D                                   for
Publisher = "Agilent VEE", Version
public class ExpressionGraphEvo : IExtensibleVeeObject
{
    GraphControl m_graph;
    List<EvoTerminal> m_inputTerminals;
```

interface Agilent.Vee.Extensibility.IExtensibleVeeObject
Represents a VEE object written by a third party.

6 EVO-related Intellisense in Step 5 should appear automatically. If it doesn't, verify **Agilent.Vee.AddInView.xml** exists in **{VEE Installed Directory}\AddInViews folder.**

Typically it is **C:\Program Files\Agilent\VEE Pro 9.3\AddInViews.**
In 64-bit OS, it could be **C:\Program Files (x86)\Agilent\VEE Pro 9.3\AddInViews.**

The VEE 9.3 installer will put the XML file inside the folder mentioned.

7 Implement the method appropriately as in **ExpressionGraphEvo.cs.**
In the expression graph solution some other classes are also implemented.

8 Close all VEE instances to allow VEE to load the newly added EVO. Build the project. After it is successfully built, copy **ExpressionGraph.evo** to either location below. (Create the folder manually if it does not exist.)

For EVO to be visible to only {user}: **C:\Documents and Settings\{user}\My Documents\Agilent\Agilent VEE 9.3\Addins**

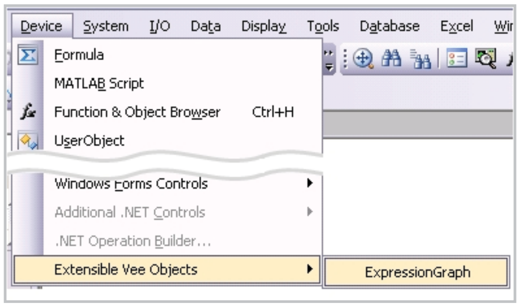
For EVO to be visible to all users logged in to the PC:
C:\Documents and Settings\All Users\Agilent\Agilent VEE 9.3\Addins

Refer to Table 1: EVO configuration file locations for other OS directories.

9 Copy the DLL and its related references to the path mentioned in the **ExpressionGraph.evo <Assembly>** field.

For example, in the example provided, **<Assembly>ExpressionGraph.dll</Assembly>** requires **ExpressionGraph.dll** and **antlr.runtime.dll** to be in the same folder as **ExpressionGraph.evo**. However, you may specify another absolute path, such as **<Assembly>C:\MyAddin\ExpressionGraph.dll</Assembly>.**

10 Launch VEE. The EVO should appears in **Device > Extensible Vee Objects** menu.



11 Drag-and-drop the EVO to VEE and it is ready to use.

Deployment

EVO assembly path

If your EVO assemblies always reside in a fixed and common directory such as **C:\Documents and Settings\All Users\Application Data\MyEVO**, then you may skip this section.

When saving an Agilent VEE program that contains an EVO, the EVO's absolute path is saved, similar to .NET DLL. (Unfortunately, VEE is currently unable to load EVOs from Global Assembly Cache (GAC).) To illustrate, let's use a scenario where an EVO assembly is located in Path A. When it is used in a VEE program, upon being saved, the EVO's full path is saved, i.e. Path A. When opening the program in another PC, if the EVO assembly is available in Path B instead of Path A, an error occurs during the loading of the EVO. This applies when running either a VEE program or VEE runtime program.

To fix the issue, the quickest method is to manually create Path A in the new machine and place the EVO assembly in Path A accordingly. If this workaround is acceptable, you may skip to next section. If this workaround is not acceptable, implementing the following deployment practices is recommended.

During VEE 9.3 installation, two groups of three system environment variables (SEVs) listed below will be added. As with the SEV nomenclature, PRO is reserved for VEE Pro, STU for VEE Student, while RUN is for VEE Runtime.

Group A: SEV for EVO file path

VEE_PRO93_EVO_PATH
VEE_STU93_EVO_PATH
VEE_RUN93_EVO_PATH

Group B: EVO SEV counter

NUM_OF_PRO93_EVO
NUM_OF_STU93_EVO
NUM_OF_RUN93_EVO

For Agilent VEE 9.2, different sets of SEV are used and those are not discussed here.

EVO assembly path *(continued)*

Table 2. Installing EVO

Step	Action
1	Check that the SEV you need to use is in Group A and Group B, if not, create it. For example, to install SEV into VEE runtime, verify that you have SEV VEE_RUN93_EVO_PATH.
2	Add your EVO file path value to the appropriate SEV value. For example: VEE_RUN93_EVO_PATH = "C:\Program Files\Agilent\VEE Pro 9.3\Lib\Toolboxes; C:\Program Files\myEvo"
3	At the appropriate SEV counter, increment the existing value by one. For example, vTools installer has increment NUM_OF_RUN93_EVO to 1. During myEVO's EVO1 installation, the value will be increased to 2.

When VEE program loads, it first uses the EVO .NET's absolute path saved in the VEE program. If VEE cannot find the DLL, it looks for the appropriate value captured in the associated SEV path.

Table 3. Un-installing EVO

Step	Action
A	Remove your path value from the SEV path in Step 2 of the prior section.
B	Decrement the SEV counter in Step 3 of the prior section.
C	Check the value of SEV counter in Step B above. If it is zero remove it. Otherwise, keep it in the system.

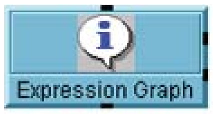
The operation above provides seamless EVO integration and VEE experience for the following:

1. Installing EVO then VEE, or
2. Installing VEE then EVO, or
3. During development, install EVO in a specific folder. However during runtime, install EVO in a different folder.

Recommended Practices

EVO icon

In the EVO configuration file (.evo), the EVO developer can specify the EVO icon be minimized by adding: **<Icon>myIcon\info.gif</Icon>**



There are two choices for the icon path value:

1. A subfolder inside the VEE bitmap folder: **{VEE Installed Directory}\bitmaps.** The icon value looks like: **<Icon>myIcon\info.gif</Icon>**

   In this example, the icon image file is: **C:\Program Files\Agilent\VEE Pro 9.3\bitmaps\myIcon\info.gif.** In 64-bit OS, it could be **C:\Program Files (x86)\Agilent\VEE Pro 9.3\bitmaps\myIcon\info.gif**

2. Use the absolute icon's full path, for example: **<Icon>C:\My product\myIcon\info.gif</Icon>**
   When using the full absolute path and deploying the program to another PC, remember that the image needs to reside in the same full absolute path. If there is a path mismatch, when the user minimizes the VEE object, an icon not found message will appear.

EVO assemblies path location

In the EVO configuration file (.evo), the EVO developer can specify the EVO assembly location by using:
<Assembly>ExpressionGraph.dll</Assembly>

There are two choices for the assembly path value:

1. Specifying the assembly name only: **<Assembly>ExpressionGraph.dll</Assembly>** In this case, the assembly is assumed to be in the same path as the EVO configuration file, in other words if you are using the Windows XP operating system it will be: **C:\Documents and Settings\{user}\My Documents\Agilent\Agilent VEE 9.3\Addins**. Alternatively, if you are using a Windows Vista or Windows 7 operating system it will be: **C:\Users\{user}\Documents\Agilent\Agilent VEE 9.3\Addins.**

   Please refer to the How EVO Works in VEE section of this document for alternatives on the EVO configuration file path. Using this definition, if there are a few EVOs installed, the EVO configuration file path could be full of .evo files and assemblies. This could eventually become wieldy.

2. Using the absolute assembly's full path, such as: **<Assembly>C:\MyEvo1\ExpressionGraph.dll</Assembly>**
   When the full absolute path is used and the need arises to use the program on another PC, remember that the assembly needs to reside in the same full absolute path. Refer to the EVO Assembly Path section of this document for details on how to fix relevant deployment issues.

Avoid duplicate terminal names

Duplicate names should be avoided within the same type of input type. For example, it is safe to have an input terminal named "Data" and an output terminal named "Data", but having two input terminals both named "Data" is not recommended.

If there are duplicated names, there will be no compilation error/warning in Visual Studio, however, when using the EVO in Agilent VEE, the data container value and execution may result in incorrect values.

VEE EVO terminals are case sensitive. For example if there are two data input terminals named *In1* and *in1*, VEE EVO treats these as different terminals. This is the same for data output terminals. If there is a data output terminal named *error*, it will be unique from the VEE error output terminal. However, this naming practice is not recommended because most of the other VEE objects (such as Formula) terminal names are not case sensitive. In other words, *In1* and *in1* are the same for a Formula object. Thus avoiding the use of terminal names that differ only by case sensitivity is strongly recommended.

Output terminal name vs. error output terminal

When defining an output terminal name, avoid using the name Error. Output terminal name Error is reserved. If you name an output terminal Error, there is no compilation error/warning in Visual Studio, however, when you execute the VEE program, a malfunction may occur.

Dynamic terminal

Dynamic terminal is useful during VEE program development, though it is recommended that its use be avoided when running a VEE program. This is because the program may break some terminal connections and cause unconnected pin errors. The EVO developer may hide/disable the dynamic terminal changes using the PreRun method and then show/enable it using the PostRun method.

When using dynamic terminal:
1. Avoid duplicate terminal names as mentioned in the Avoid Duplicate Terminal Names section of this document.
2. Use **SetObjectData/GetObjectData** to handle device terminal status. This allows previously set terminal information to be loaded when a user reopens the VEE program. **SetObjectData/GetObjectData** is useful in storing and retrieving EVO state information from saved VEE program file.

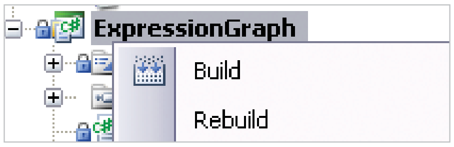
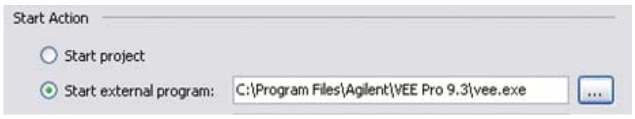
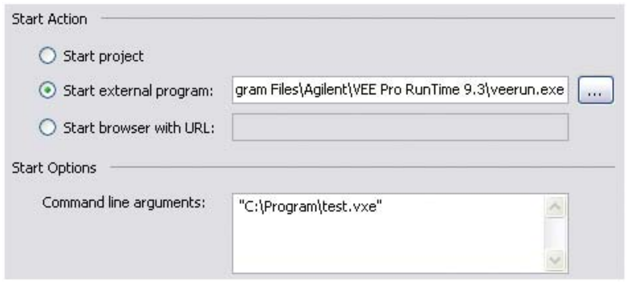
How Do I Debug EVO Code?

Regardless of whether the EVO is added from a **Device > Extensible Vee Objects** menu, or via the VEE customized menu, the debugging steps are identical. The same debugging steps are applicable for running a VEE runtime program containing EVO.

There are two ways to debug EVO .NET source code.

1. Launch the VEE instance from Visual Studio. EVO is a Class Library project type, thus it is not allowed to be started in Visual Studio when clicking on **Start Debugging**. Use the following procedure to debug EVO .NET source code:

Table 4. Debugging

Step	Action
1	In Visual Studio **Solution Explorer**, right click on the Project. 
2	Select **Properties**.
3	In **Debug > Start Action**, select **Start external program** and point to the VEE Pro 9.3 executable file.  Start Action Start project Start external program: C:\Program Files\Agilent\VEE Pro 9.3\vee.exe If you are debugging from a VEE runtime program (.vxe), point the start up program to the VEE Runtime 9.3 executable file and enter the runtime program name in the **Command line arguments**.  Start Action Start project Start external program: gram Files\Agilent\VEE Pro RunTime 9.3\veerun.exe Start browser with URL: Start Options Command line arguments: "C:\Program\test.vxe"
4.	Click **Start Debugging** or **F5**. Visual Studio brings up a VEE Pro (or VEE Runtime) instance. Use the EVO in VEE. If you put a break point in the EVO source code, it breaks accordingly and you may debug your EVO source code now.

2. Go to the Visual Studio menu **Debug > Attach to Process**. Select the VEE Pro (or VEE runtime) instance containing the EVO instance.

If there is error in loading EVO, please refer to the EVO Assembly Path section of this document.

Some Notes on the Expression Graph Example

If there is compilation error about a missing Agilent.Vee.AddInView.dll, add the DLL reference following the steps below:

Table 5. Adding the DLL reference

Step	Action
1	Remove the **Agilent.Vee.AddInView** reference. Right click on **Agilent.Vee.AddInView**. Select Remove.
2	Add a reference to: **Agilent.Vee.AddInView in {VEE Installed Directory}\AddInViews.** Typically it is: **C:\Program Files\Agilent\VEE Pro 9.3\AddInViews\Agilent.Vee.AddInView.dll.** In 64-bit OS, it could be: **C:\Program Files (x86)\Agilent\VEE Pro 9.3\AddInViews\Agilent.Vee.AddInView.dll**
3	Rebuild the solution. No compilation error should occur. Please contact Agilent for support if other difficulties are experienced.

Appendix A: Agilent.Vee.Extensibility.IExtensibleVeeObject Interface

This table lists all members in IExtensibleVeeObject interface with description.

Table 6. Properties

	Name	Description
	Title	Gets the default title of EVO. User can change it in VEE.
	ControlInputs	Gets the list of EVO control input terminals. It serves as the optional data input terminals. Its behavior is same as the control input terminals of the original VEE objects. Avoid using control input terminals names if possible.
	DataInputs	Gets the list of EVO data input terminals.
	DataOutputs	Gets the list of EVO data output terminals.
	DefaultWidth	Gets the default width of the EVO.
	DefaultHeight	Gets the default height of the EVO.
	ErrorOutput	Gets/Sets the EVO error message. If the EVO error output terminal is added, it will be stored in that terminal; otherwise VEE will display the message in runtime error dialog.

Table 7. Methods

	Name	Description
	GetControl	The UI control of the Extensible VEE Object. It can be Windows Forms Control or WPF window. Returns null if there is no EVO UI.
	PreRun	This method executes before the VEE program starts to run.
	Execute	This method executes when the VEE program executes to this EVO. If there is an unhandled exception in this method, VEE catches it and treats it as a runtime error.
	PostRun	This method executes right after the Execute method and before the VEE program ends.
	GetCurrentApplication	Gets a reference to the current instance of the Agilent VEE application, which is the root object of the Agilent VEE automation model. It is a good practice to check that the application is not null before using it.
	GetObjectData	Gets the data representing the EVO state. VEE saves the data into the VEE program file.
	SetObjectData	When opening a saved VEE program, saved EVO state data is serialized into a string. The string data is passed to EVO to process the state.
	ShowHelp	Launches the EVO help function.
	InputEvoTerminalAdded	This method executes after the EVO input terminal is added.
	InputEvoTerminalRemoved	This method executes after the EVO input terminal is removed.
	ControlEvoTerminalAdded	This method executes after the EVO control terminal is added.
	ControlEvoTerminalRemoved	This method executes after the EVO control terminal is removed.

Table 8. Events

	Name	Description
	ModifiedSet	An event to notify the VEE that there are EVO changes. When this event is fired, the VEE displays an asterisk symbol in the program title bar.
	TerminalChanged	An event to notify the VEE that there is a change in input/output/control terminal.

Appendix B: EVO Configuration File (.evo)

This is the layout and description of the content of EVO configuration file (.evo).

```
<?xml version="1.0" encoding="UTF-16" standalone="no"?>
<Extensibility xmlns="http://www.agilent.com/Schemas/AutomationExtensibility">
  <HostApplication>
    <Name>Agilent VEE</Name>
    <!-- Change to appropriate VEE version, i.e. 9.3 -->
    <Version>9.3</Version>
  </HostApplication>
  <Addin>
    <!-- EVO default name displays in Device menu -->
    <FriendlyName>ExpressionGraph</FriendlyName>

    <!-- Text in status bar when the EVO menu item is highlighted in VEE -->
    <Description>Realtime expression graph</Description>

    <!-- The full path of the EVO assembly -->
    <!-- In the example below, VEE will search ExpressionGraph.dll in
Addins folder -->
    <!-- If full absolute path e.g. C:\MyProject\ExpressionGraph.dll is
specified, VEE will search for the assembly in the path specified. -->
    <Assembly>ExpressionGraph.dll</Assembly>

    <!-- The full class name of the EVO -->
    <FullClassName>Agilent.Vee.EvoExample.ExpressionGraphEvo</FullClassName>

    <!-- Currently not in use, leave it as 0 -->
    <StartUp>0</StartUp>

    <!-- Display icon picture when EVO is minimized in VEE -->
    <!-- In example below, VEE will look for info.gif in VeeInstallDir\bitmaps\
myIcon folder-->
    <!-- If full absolute path is specified e.g. C:\MyIcon\info.gif, VEE will
search for the icon in the path specified. -->
    <Icon>myIcon\info.gif</Icon>

    <!-- Set to 1,to add EVO into the Device menu. If 0, the EVO will not be
visible -->
    <CreateMenu>1</CreateMenu>
  </Addin>
</Extensibility>
```

www.agilent.com
www.agilent.com/find/vee

Agilent Email Updates

www.agilent.com/find/emailupdates
Get the latest information on the products and applications you select.

www.lxistandard.org
LAN eXtensions for Instruments puts the power of Ethernet and the Web inside your test systems. Agilent is a founding member of the LXI consortium.

Agilent Channel Partners

www.agilent.com/find/channelpartners
Get the best of both worlds: Agilent's measurement expertise and product breadth, combined with channel partner convenience.

Agilent Advantage Services

Agilent Advantage Services is committed to your success throughout your equipment's lifetime. To keep you competitive, we continually invest in tools and processes that speed up calibration and repair and reduce your cost of ownership. You can also use Infoline Web Services to manage equipment and services more effectively. By sharing our measurement and service expertise, we help you create the products that change our world.

www.agilent.com/find/advantageservices

Agilent Electronic Measurement Group
DEKRA Certified
ISO 9001:2008
Quality Management System
www.agilent.com/quality

For more information on Agilent Technologies' products, applications or services, please contact your local Agilent office. The complete list is available at:

www.agilent.com/find/contactus

Americas	
Canada	(877) 894 4414
Brazil	(11) 4197 3600
Mexico	01800 5064 800
United States	(800) 829 4444

Asia Pacific	
Australia	1 800 629 485
China	800 810 0189
Hong Kong	800 938 693
India	1 800 112 929
Japan	0120 (421) 345
Korea	080 769 0800
Malaysia	1 800 888 848
Singapore	1 800 375 8100
Taiwan	0800 047 866
Other AP Countries	(65) 375 8100

Europe & Middle East	
Belgium	32 (0) 2 404 93 40
Denmark	45 45 80 12 15
Finland	358 (0) 10 855 2100
France	0825 010 700*
	*0.125 €/minute
Germany	49 (0) 7031 464 6333
Ireland	1890 924 204
Israel	972-3-9288-504/544
Italy	39 02 92 60 8484
Netherlands	31 (0) 20 547 2111
Spain	34 (91) 631 3300
Sweden	0200-88 22 55
United Kingdom	44 (0) 118 972 6201

For other unlisted countries:
www.agilent.com/find/contactus

Revised: January 6, 2012

Product specifications and descriptions in this document subject to change without notice.

© Agilent Technologies, Inc. 2012
Published in USA, January 23, 2012
5990-9437EN

Agilent Technologies