Visual Basic - Technical notes and limitations

Summary: This section provides more detail about the support for Visual Basic and the way in which it is supported.

Referencing through UNC path

A DLL Project with a local path is not matched when referenced through UNC path. When VB project "A" references a DLL "B" through a UNC path and the VB project "B" corresponding to the referenced DLL "B" has a local path using a drive letter, the VB analyzer does not match DLL "B" with project "B" and therefore does not create links between "A" and "B".

Integers

The VB analyzer's Inference Engine does not take integers into account. When an integer participates in the construction of a string (strC1 = "Inference. clsVB" + CStr(i)), the Inference Engine does not find the true value. In the previous example, it finds only "Inference.clsVB" and not "Inference.clsVB1"," Inference.clsVB2".

Unresolved OCX Method Calls

Calls to the methods listed below are not detected by the VB Analyzer. These methods are dynamically added to OCA files the first time Visual Basic is launched. These methods only exist for graphical components contained in OCXs.

Method Name	Description
BSTR Name()	Returns the name used in code to identify an object.
short index()	Returns/sets the number identifying a control in a control array.
single Left()	Returns/sets the distance between the internal left edge of an object and the left edge of its container.
void Left([in] single rhs)	Returns/sets the distance between the internal left edge of an object and the left edge of its container.
single Top()	Returns/sets the distance between the internal top edge of an object and the top edge of its container.
void Top([in] single rhs)	Returns/sets the distance between the internal top edge of an object and the top edge of its container.
single Width()	Returns/sets the width of an object.
void Width([in] single rhs)	Returns/sets the width of an object.
single Height()	Returns/sets the height of an object.
void Height([in] single rhs)	Returns/sets the height of an object.
VARIANT_BOOL Visible()	Returns/sets a value that determines whether an object is visible or hidden.
void Visible([in] VARIANT_BOOL rhs)	Returns/sets a value that determines whether an object is visible or hidden.
IDispatch* Parent()	Returns the object on which this object is located.
short DragMode()	Returns/sets a value that determines whether manual or automatic drag mode is used.
void DragMode([in] short rhs)	Returns/sets a value that determines whether manual or automatic drag mode is used.
Picture* DragIcon()	Returns/sets the icon to be displayed as the pointer in a drag-and-drop operation.
void DragIcon([in] Picture* rhs)	Returns/sets the icon to be displayed as the pointer in a drag-and-drop operation.

void DragIcon([in] Picture* rhs)	Returns/sets the icon to be displayed as the pointer in a drag-and-drop operation.
BSTR Tag()	Stores any extra data needed for your program.
void Tag([in] BSTR rhs)	Stores any extra data needed for your program.
VARIANT_BOOL Enabled()	Returns/sets a value that determines whether an object can respond to user-generated events.
void Enabled([in] VARIANT_BOOL rhs)	Returns/sets a value that determines whether an object can respond to user-generated events.
VARIANT_BOOL TabStop()	Returns/sets a value indicating whether a user can use the TAB key to give the focus to an object.
void TabStop([in] VARIANT_BOOL rhs)	Returns/sets a value indicating whether a user can use the TAB key to give the focus to an object.
short TabIndex()	Returns/sets the tab order of an object within its parent form.
void TabIndex([in] short rhs)	Returns/sets the tab order of an object within its parent form.
IDispatch* Object()	Returns an object in a control.
long HelpContextID()	Specifies the default Help file context ID for an object.
void HelpContextID([in] long rhs)	Specifies the default Help file context ID for an object.
long WhatsThisHelpID()	Returns/sets an associated context number for an object.
void WhatsThisHelpID([in] long rhs)	Returns/sets an associated context number for an object.
IDispatch* Container()	Returns the container of an object.
void Container([in] IDispatch* rhs)	Returns the container of an object.
VARIANT_BOOL CausesValidation()	Returns/sets whether validation occurs on the control which lost focus.
void CausesValidation([in] VARIANT_BOOL rhs)	Returns/sets whether validation occurs on the control which lost focus.
DataBindings* DataBindings()	Returns/sets a DataBindings collection object that collects the bindable properties that are available to the developer.
BSTR ToolTipText()	Returns/sets the text displayed when the mouse is paused over the control.
void ToolTipText([in] BSTR rhs)	Returns/sets the text displayed when the mouse is paused over the control.
void SetFocus()	Moves the focus to the specified object.
void ZOrder([in] VARIANT Position)	Places a specified object at the front or back of the z-order within its graphical level.
void Move([in] single Left, [in] VARIANT Top, [in] VARIANT Width, [in] VARIANT Height)	Moves an object.
void Drag([in] VARIANT Action)	Begins, ends, or cancels a drag operation of any object except Line, Menu, Shape, and Timer.
void ShowWhatsThis()	Displays a selected topic in a Help file using the What's This popup provided by Windows 95 Help.

Miscellaneous issues

- The VB analyzer searches for server object names in all literal strings and embedded SQL in Visual Basic, taking into account normalized. While this technique searches for and finds all real schema/database object references, it may also find incorrect references. For example, literal strings, such as user messages or menu options, may contain the names of database or schema objects and in such cases, incorrect references may be found. The VB analyzer will NOT search for, NOR find the names of stored procedures, tables or views that are constructed dynamically by concatenating parts of their names.
- The VB analyzer does not scan binary files, as a result, it will NOT detect references hidden in these files.
- If you declare a generic object variable (such as an object) and assign an object to it by using the Set statement, the VB analyzer will NOT detect references to a method of this object.
- Collections and arrays are not supported during resolution via these object types. In other words, these objects can be seen in the Graphical View, but when a call is made via a collection or an array, the elements located on the right of the call are not found: *MyClass.MyMethod()* is fully supported, but in *MyArrayOfClass(n).MyMethod()* no link will be detected to *MyClass* nor to *MyMethod.* A link will, however, be detected to *MyArr ayOfClass.* This limitation only applies for collections or variant/object arrays. For arrays whose type is already known when the array is declared, the link to the method will be detected.
- Control groups are supported as follows: only the group is displayed by CAST - each element in the group cannot be seen.
- Win32 calls from the VB code are represented by imported functions.
- Binary files are not analyzed.

Inference Engine limitations

Because of a limitation in the Inference Engine for the VB analyzer, the link between "sub1" and the method "Method" (in the example below) will not be created where a parameter of a parametrized method ("MsgBox" in this case) calls a dynamic function (i.e. a link retrieved by the Inference Engine). For example: *Sub Sub1()Dim var as ObjectSet o = CreateObject("A.B")MsgBox o.Method()End Sub*

Links from VB objects to COM objects

Links to COM methods implemented in an OCX library or C++ DLL are not created while analyzing VB source code having references to these methods. these links can, however, be created manually using a Reference Pattern.

Syntax

When the line continuation character, i.e. the underscore (_), is used after a period (.), the VB analyzer gives a syntax error during analysis of such code. Example:

```
frmToursPerdusManqueConducteur.flexDonnees. _
                                      TextMatrix(ii, 7) = vf_NbTPJour
--------------------------------------
```

As a consequence, the object containing this code is not analyzed. To avoid the error, either remove the line continuation character and write the code on one line, or separate the line at a different location, not after a period. The use of the line continuation character after other characters than the period does not cause a syntax error.

VB Standard APIs

Standard VB APIs (VBA, VB, VBRUN) delivered with the various versions of Visual Basic are too big to be analyzed during each analysis. Consequently these APIs have been pre-parsed and the underlying classes are packaged with the analyzer. The VB APIs are then loaded depending on the version and redeployed on demand during the analysis.

This has the following consequences:

- VB standard objects are not stored in the Analysis Service.
- Even if references to standard VB objects are resolved by the analyzer, corresponding links are not traced and consequently not stored.
- For the **CreateObject** method and **CreateInstance** method, the analyzer will create a link between the caller of these methods and the COM CoClass of the first parameter of these methods.