VCF1CMPT.WRI

This file describes the compatibility layer VCF1CMPT.TXT which will help you move legacy code to the OCX environment. The compatiblity layer provides a simple mapping of functions used in the VBX products syntax to the OCX syntax. Some functions that were present in the VBX were VBX specific and are not present in the OCX. There are a few modifications you will have to make to your code which are described below. There will be no DLL type interface for the OCX and hence none of the new functionality will be supported. In general, the OCX is faster than the VBX because the OCX is more tightly coupled to the calculation engine (they are in the same file). Operating system enhancements and improvements to the Formula One engine will increase this speed advantage.

The first step to migrating your VBX projects to the OCX environment is to save your project in text form. This is a requirement for any project to be moved from VB3 to VB4. When in text form, your .FRM files can be opened in a text processor. You need to open each form and use a global search and replace, changing "SSView" to "F1Book1". This step will preserve all of your custom formatting and property settings. You should work on a copy of your project so that if an error occurs, no work is lost.

Remove the Formula One declarations file from your project. This file is named "VTSS.TXT" or "VTSS.BAS" in the shipping version. Add this file (VCF1CMPT.TXT) to your project.

During the installation process, a line was added to your VB.INI that maps your VBX to the OCX. When you open your VB3 project in VB4, you will be asked if you want to upgrade a list of controls. VTSS.VBX will be on that list.

The spreadsheet handle is not required in most OCX methods. The compatibility layer needs to identify the Formula One control to operate on so it uses the object passed by reference instead. To easily change from your VB3 syntax that used a spreadsheet handle, use a global search and replace on <FO Control>.SS and remove the ".SS". For example, if your control name was "ssFinancials", then set "Find What" to "ssFinancials.SS" and set "Replace With" to "ssFinancials".

The next step is to change any Formula One constants that you used to their new names. For example, kFileFormulaOne should be changed to F1FileFormulaOne. You can use your help file or the VB4 browser to findout the new names.

For most projects, that's it, now you are OCX powered - although you still have a layer of indirection. There are a few functions that need special attention. See the list and and instructions at the bottom of this discussion.

What if my function does not work?

Review the special handling section at the bototm of this discussion. If the answer is not there, look for the definition of your function in this section. The function definition in this file is how the function works in compatibility mode. If you can't figure the problem out, just recode it as an OCX method or property.

How does it this compatibility thing work?

The compatibility layer simply maps the DLL calls to OCX calls.You are able to use nearly the same syntax as with your VBX projects. The Compatibility layer contains functions with the same name as the Formula One VBX functions. Now when you call something like SSAddColPageBreak, the compatibility layer function SSAddColPageBreak is called and the OCX method is called. If an error occurs, it is trapped and its number is returned as the function value. The Formula One Control is passed by reference for a speed enhancement - it is not modified in any of the code below.

How do I migrate?

If your program seems sluggish, you will want to start your OCX migration right away. The sluggishness is probably due to the extra code in the compatibility layer. Our testing shows the OCX to have the same or much better performance than the VBX/DLL combination.

Remember: Make the common case fast. You want to find all the tight loops where many calls are made to Formula One in a rapid succession. VB4 provides a code profilier to help you identify where most of the time is spent when your code is executing. See VB docs for more info.

The migration is simply replacing your old syntax with the OCX syntax in your function calls. There should be minimal logic changes except for error handling. You no longer have to check the return value for each function call, just make an error handler for the errors you want to catch.

It is a good idea to put an error handler in each Function or Sub-Procedure that calls Formula One. This will prevent a run time error that is unrecoverable. Although the errors may be highly unlikely, your users will want to see some error dialog instead of a can't continue message. These are the same type of error handlers as in VB3. They have a "On Error Goto ErrorHandler" statement at the top and a "<ErrorHandler>:" statement somewhere in the function or sub-procedure.

Functions not supported:

SSCallWindowProc, SSEditBarDelete/Move/New, SSNew, SSDelete, SSSetDefWindowProc: Although defined for the VBX, a warning appeared in the manual not to use them and they are no longer supported.
SSCanEditPaste: Not supported in this version of Formula One.
SSMaxRow/Col: Not supported in this version.
SSObjNewPolygon, SSObjSetPolygonPoints: Do not apply to this version of Formula One.
SSEditBarHeight, SSGet/SetSSEdit: Are not available - they do not apply to this version of Formula One.
SSVBXCopyCellsFrom/ToDoubleArray: Applies to VBX only. Not provided in this version of Formula One.
SSVBXRead/WriteBasicFile: Applies to VBX only. Not provided in this version of Formula One.

Functionality changes:

SSCheckModified: Replaced by the Modified property.
SSGet/SetDoSetCursor: Use the new MousePointer and MouseIcon properties instead.
SSGet/SetFireEvent: Use the Do* properties instead (DoClick, etc.)
SSRead/WriteIO: Replaced by the new ReadFrom/WriteToBlob methods.

Functions whose argument count have changed for the compatibility layer:

SSErrorNumberToText: Now takes an additional first argument, a Formula One control as an Object.
SSUpdate - now needs 1 argument, the Formula One control to update.
Declaration -> Public Sub SSUpdate (SS as Object)
SSVersion - now needs 1 argument, the Formula One control.
Declaration -> Public Function SSVersion (SS as Object)

Other Changes:

Row and Column references are now longs instead of integers. Since the maximum row and column values fit in a integer data type, this should not cause problems unless you use variables. VB will catch the type disagreement - you just need to change the type.