Debugging Release Scripts Created Using Visual Basic 6

Application Note

Date	August 24, 2011
Applies To	Kofax Capture 8.0, 9.0, and 10.0
Summary	This application note provides a technique for debugging Visual Basic 6 Release Scripts within the Release Process.
Revision	1.1

Overview

When developing and debugging Release Scripts, it is often desirable to step through the code while a document is being released.

This provides invaluable insight into what is actually happening 'under the hood', and can help in determining if there is a logic issue with a Release Script's implementation, or the exact nature of an error.

This Application Note, using the Kofax Capture Text Release Script, illustrates some common techniques for debugging a Release Script created using Visual Basic 6.

Compiling with Binary Compatibility

When debugging a Release Script, it is often required to recompile the Release Script as part of the process.

To ensure that no incompatibilities are introduced between the newly compiled Release Script and the one that is currently installed, it is recommended to set the Source Code project for Binary Compatibility.

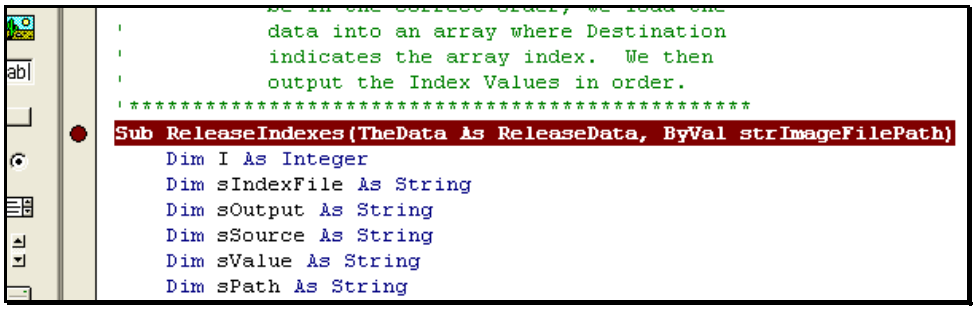
Stepping through the Code

In order to see what is happening with a Release Script, it is often necessary to attach to the Release process and step through the code in the Visual Basic Debugger.

To do this, it is best to isolate a portion of the Source Code, and place a breakpoint.

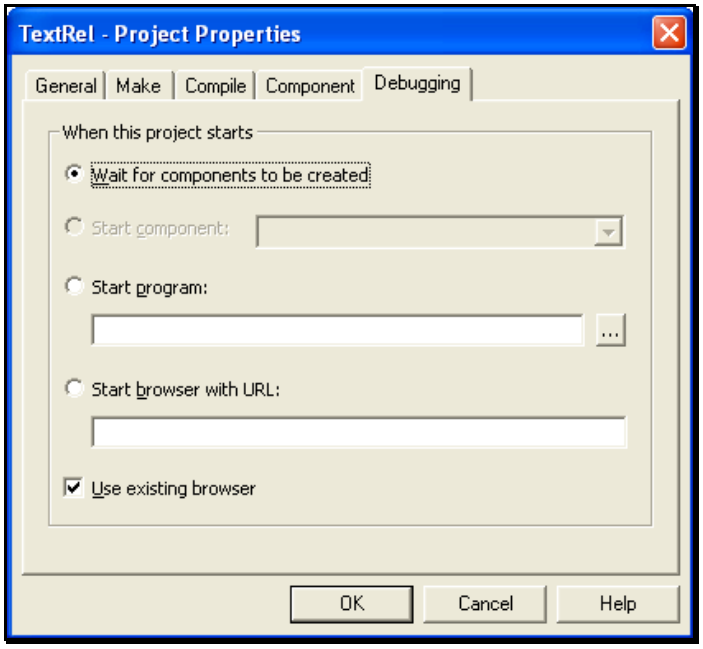
First, go to the appropriate line of code, for example the **ReleaseIndexes** subroutine in the ASCIITextFile class (Text.cls), which appends selected Index Field values to the selected text file.

Then, press **F9** to set the Breakpoint:



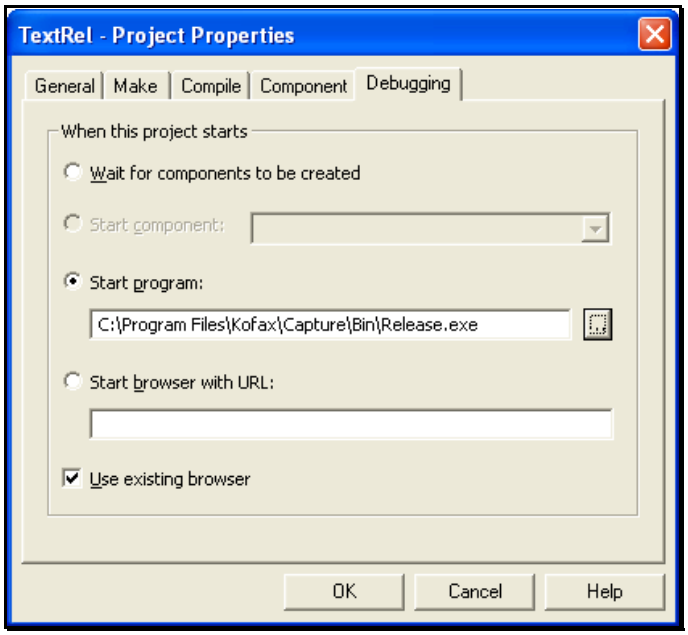
Next, we need to configure Visual Basic to run the Release module for Debugging.

To do this, go to the **Project → Properties** menu item, then go to the **Debugging** tab:

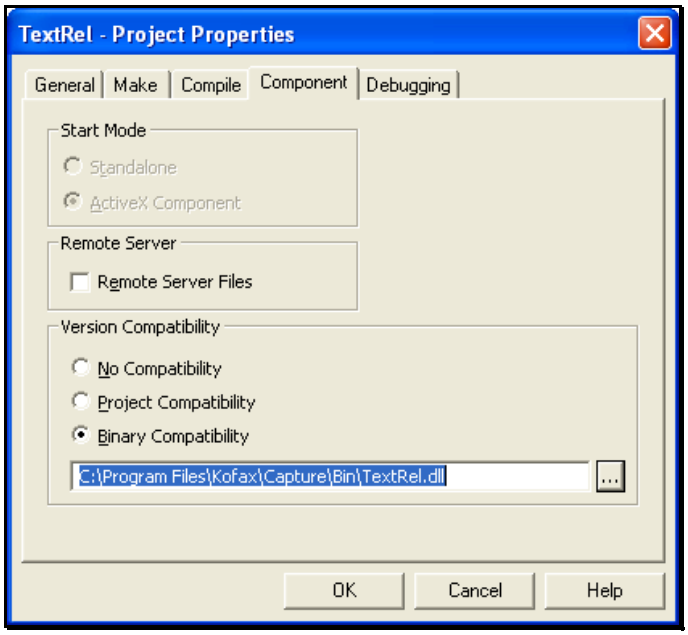


Select the **Start program** option, then select the ellipsis (...) to open the **Launch Program** dialog. From here, select the Capture\BIN folder, find **Release.exe**, then click **Open**.

The screen should look similar to this:



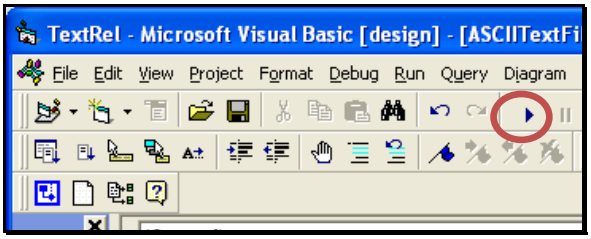
On the **Component** tab, in the **Version compatibility** section, select **Binary Compatibility**, then click the ellipsis and open TextRel.dll in the Capture\BIN folder:



Now click OK, and Visual Basic is ready to debug.

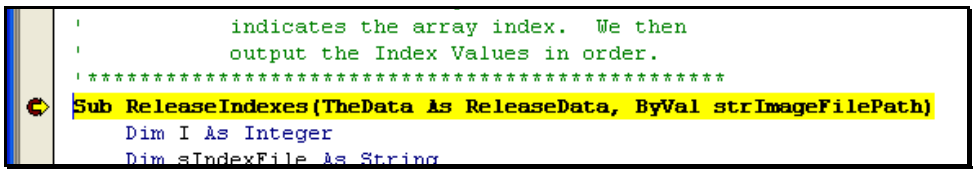
Next, process a Batch that references the Release Script all the way to the Release queue. This way, the Batch will be opened by the Release module when run.

Next, start the Debug process from the Visual Basic IDE, by clicking on the Start Debug button:



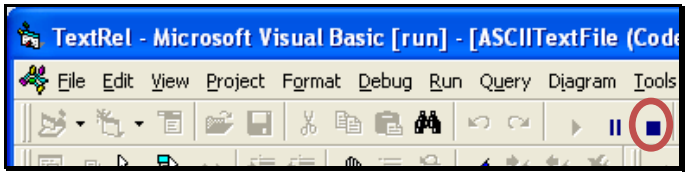
The Release Module will then launch, and call the Release Script to release the Document.

When the Release Script process reaches the Breakpoint defined earlier, Release will stop, and the Visual Basic IDE will highlight the currently running line of code:



Now press the **F8** key to move to the next step in the code. Each time the **F8** key is pressed; one operation will be performed, revealing exactly what code is being run. To continue processing normally, simply press the **F5** key, and Release will continue to process until the next time a Breakpoint is encountered.

To stop debugging, simply click the **Stop** button:



Conclusion

Using the basic technique highlighted here, it is possible to ascertain exactly what is happening within a Release Script, and to possibly formulate a change to avoid encountering the error condition, or alter the Release Script to handle a given issue differently.