Thank you for your interest in VersionStamper. This edition of VersionStamper is a fully functional 32 bit OLE control. The only limitation is that it will not run inside of an executable. You may run use this demo control only from within vb32.exe. You must first register the control by selecting the VB "Tools - Custom Controls" menu command and adding the "Desaware VersionStamper Control Demo".

Please refer to the VerDem32.exe and VerSamp.vbp for an example on how to use VersionStamper. There are many ways that VersionStamper can be used to detect file conflicts, these demos only show a couple of different methods. The remainder of this file describes some of commonly used properties, methods and events.

Properties

Properties Available During Scanning and Verification

The following properties can be used during the FileConflict and FileScan events to obtain information about the files and conditions that triggered the event. Most of these properties are accessed through the functions found in the VerVrfy.bas and English.bas files.

FoundDate Property
This property can be used during the FileConflict event to obtain the file date for the file which triggered the conflict warning for this event. The date is retrieved in the form of a Long variable that represents the number of seconds that have elapsed since midnight, December 31, 1899 UTC. This property is typically used in the Visual C++ environment which counts time from this date. Visual Basic programmers will typically use the FoundDateString property.

This property is only valid during the FileConflict event and is read only.

FoundDateString Property
This property can be used during the FileConflict event to obtain the file date for the file which triggered the conflict warning for this event. The date is returned as a string using the short form date and time notation.
The format for this string takes into account the internationalization parameters set by the Windows environment. Visual Basic will be able to correctly interpret this string, however you should be careful to use the current internationalization parameters if you parse the string yourself. Visual C++ programmers will typically use the FoundDate property which returns the date as a Long variable.

This property is only valid during the FileConflict event and is read only.

FoundFlags Property
This property can be used during the FileConflict events to obtain the FileFlags for the file which triggered the conflict warning for this event. This property is only valid during the FileConflict and is read only. The property type is Long.

FoundSize Property (OCX edition only)
This property can be used during the FileConflict event to obtain the file size for the file which triggered the conflict warning for this event. The size is retrieved in the form of a Long variable that represents the number of bytes.
This property is only valid during the FileConflict event and is read only.

FoundVersion Property
This property can be used during the FileConflict event to obtain the version number for the file which triggered the conflict warning for this event. This property is only valid during this event and

is read only.
This version number is returned as a string that has from one to four numbers separated by periods in the form:
A.B.C.D
The first two numbers (A and B) are typically the major and minor version numbers (for example: version 1.0). C is typically used to identify maintenance releases, and D to identify internal releases.

Each number can range from zero to 32767.

FoundVersionBuffer Property
This property can be used during the FileConflict event to obtain a string containing a version buffer for the file date for the file which triggered the conflict warning for this event. This buffer can be used with the Version API functions to obtain detailed version information for the file.
This property is not available under Visual C++ and other environments compatible with version 1.0 of the VBX standard. These users will need to use API functions to obtain the version buffer based upon the filename.

This property is only valid during the FileConflict event and is read only.

OtherCount Property
This property can be used during the FileConflict event to determine how many other files have the same name as the reference file in the search path. It contains an integer that specifies the number of entries in the OtherFile, OtherVersion, OtherFlags, OtherDate, OtherDateString, and OtherSource property arrays.
This property is only valid during the FileConflict event and is read only.

On accessing the Other... properties. All of the Other.... properties (OtherDate, OtherDateString, OtherFile, OtherFlags, OtherSize, OtherSource and OtherVersion) are property arrays whose length depends on the number of files found and whose first entry is always index zero. If no other files were found, any attempt to read these properties will cause an Invalid Property Array Index error. Use the OtherCount property to determine the number of entries in these property arrays.

OtherDate Property
This property can be used during the FileConflict event to obtain the dates for other files that have the same name as the reference file. This is useful for determining how to correct conflict situations, as it allows you to quickly determine if the correct version of a file is anywhere else in the search path.
This is a property array whose length can be determined using the OtherCount property. The first entry in the array has the same value as the FoundDate property for this event.

The date is retrieved in the form of a Long variable that represents the number of seconds that have elapsed since midnight, December 31, 1899 UTC. This property is typically used in the Visual C++ environment which counts time from this date. Visual Basic programmers will typically use the OtherDateString property.
This property array is only valid during the FileConflict event and is read only.

OtherDateString Property
This property can be used during the FileConflict event to obtain the dates for other files that have the same name as the reference file. This is useful for determining how to correct conflict situations, as it allows you to quickly determine if the correct version of a file is anywhere else in the search path.
This is a property array whose length can be determined using the OtherCount property. The first entry in the array has the same value as the FoundDateString property for this event.

The format for this string takes into account the internationalization parameters set by the Windows environment. Visual Basic will be able to correctly interpret this string, however you should be careful to use the current internationalization parameters if you parse the string yourself. Visual C++ programmers will typically use the OtherDate property which returns the date as a Long variable.
This property array is only valid during the FileConflict event and is read only.

OtherFile Property
This property can be used during the FileConflict event to obtain a list of other files that have the same name as the reference file. This is useful for determining how to correct conflict situations, as it allows you to quickly determine if the correct version of a file is anywhere else in the search path.
This is a property array whose length can be determined using the OtherCount property. The first entry in the array is usually the same as the Found File for this event.

This property array is only valid during the FileConflict event and is read only.

The first entry in the Other... properties. The first entry in the Other.... properties during a FileConflict event will generally match the file described by the Found.... properties. The exception is when the "Found" file was found in memory. In this case it is possible that no other files were found, or that the Found file is later in the Other files list.

OtherFlags Property
This property can be used during the FileConflict event to obtain the FileFlags for other files that have the same name as the reference file. This is useful for determining how to correct conflict situations, as it allows you to quickly determine if the correct version of a file is anywhere else in the search path.
This is a property array whose length can be determined using the OtherCount property. The first entry in the array has the same value as the FoundFlags property for this event.

This property array has the Long data type, is only valid during the FileConflict event and is read only.

OtherSize Property
This property can be used during the FileConflict event to obtain the file size for other files that have the same name as the reference file. This is useful for determining how to correct conflict situations, as it allows you to quickly determine if the correct size of a file is anywhere else in the search path.
This is a property array whose length can be determined using the OtherCount property. The first entry in the array has the same value as the FoundSize property for this event. The size is retrieved in the form of a Long variable that represents the number of bytes.

This property is only valid during the FileConflict event and is read only.

OtherSource Property
This property can be used during the FileConflict event to determine the location of other files that have the same name as the reference file. This is useful for determining how to correct conflict situations, as it allows you to quickly determine if the correct version of a file is anywhere else in the search path.
This is a property array whose length can be determined using the OtherCount property.
This property is an integer that can have one of the following values:

Value	Definition
0	A conflicting module with the same name as the reference was found loaded into memory. This usually occurs when another application uses the same file, but loaded an older version of the file.

1 A conflicting module can be found in the current directory.
2 A conflicting module can be found in the Windows directory.
3 A conflicting module can be found in the System directory.
4 A conflicting module can be found in a directory that is part of the PATH environment string.
5 A conflicting module can be found in the project directory (the directory that contains the current executable).
This property array is only valid during the FileConflict event and is read only.

OtherVersion Property
This property can be used during the FileConflict event to obtain the versions of other files that have the same name as the reference file. This is useful for determining how to correct conflict situations, as it allows you to quickly determine if the correct version of a file is anywhere else in the search path.
This is a property array whose length can be determined using the OtherCount property. The first entry in the array usually has the same value as the FoundVersion property for this event.

This property array is only valid during the FileConflict event and is read only.
This version number is returned as a string that has from one to four numbers separated by periods in the form:
A.B.C.D
The first two numbers (A and B) are typically the major and minor version numbers (for example: version 1.0). C is typically used to identify maintenance releases, and D to identify internal releases.
Each number can range from zero to 32767.

ReadPropErrors Property
This property determines whether or not a runtime error is triggered for the RefDate, RefDateString, FoundDate, RefVersion, FoundVersion, FoundVersionBuffer, OtherDate, OtherDateString, and OtherVersion properties.
It is possible in many cases for date and version information to be unavailable for a particular component. For example, files that are not in an executable format will not have version resources available. With this property set to False (the default), attempting to read non existent information will simply return an empty string. When this property is set to true, attempting to read non existent information will cause a runtime error to occur.

RefDate Property
This property can be used during the FileScan and FileConflict events to obtain the file date for the reference file for the event. The date is retrieved in the form of a Long variable that represents the number of seconds that have elapsed since midnight, December 31, 1899 UTC. This property is typically used in the Visual C++ environment which counts time from this date. Visual Basic programmers will typically use the RefDateString property.

This property is only valid during the FileScan and FileConflict events and is read only.

RefDateString Property
This property can be used during the FileScan and FileConflict events to obtain the file date for the reference file for the event.
The format for this string takes into account the internationalization parameters set by the Windows environment. Visual Basic will be able to correctly interpret this string, however you should be careful to use the current internationalization parameters if you parse the string yourself. Visual C++ programmers will typically use the FoundDate property which returns the date as a Long variable.

This property is only valid during the FileScan and FileConflict events and is read only.

RefFlags Property
This property can be used during the FileScan and FileConflict events to obtain the FileFlags information for the reference file for the event. This property is only valid during these events and is read only. The property type is Long.

RefSize Property
This property can be used during the FileScan and FileConflict events to obtain the file size for the reference file. The size is retrieved in the form of a Long variable that represents the number of bytes.
This property is only valid during the FileScan and FileConflict events and is read only.

RefVersion Property
This property can be used during the FileScan and FileConflict events to obtain the version number for the reference file for the event. This property is only valid during these events and is read only.
This version number is returned as a string that has from one to four numbers separated by periods in the form:
A.B.C.D
The first two numbers (A and B) are typically the major and minor version numbers (for example: version 1.0). C is typically used to identify maintenance releases, and D to identify internal releases.

Each number can range from zero to 32767.

Events

FileConflict(ReferenceFile As String, FoundFile As String, Flags As Long, StopVerify As Integer)

VersionStamper allows you to embed version information and warning conditions for components into your Visual Basic executable. During file verification, the VersionStamper control scans through this embedded information and searches the system for the first file it encounters that matches the component name. It compares the version and date information found against the warning conditions included in the file. If any of the warning conditions are detected, a FileConflict event is triggered with the following parameters:

Parameter **Description**
ReferenceFile The embedded file name. This will include a full or relative path only if specified during the embedding process.
FoundFile The full path of the file that was found that matches the embedded file. The empty string if no file was found.
Flags A Long that determines which warnings were triggered. It can be a combination of the following:Bit 0 - Older file was found (Date compare).Bit 1 - Newer file was found (Date compare).Bit 2 - Older version found (Version compare).Bit 3 - Newer version found (Version compare).Bit 4 - Special version found.Bit 5 - Always warn was set.Bit 6 - File was found in memory.Bit 7 - No matching file was found.Bit 8 - A version compare was requested but no version resource was found in file FoundFile.Bit 9 - A Larger file was found.Bit 10 - A Smaller file was found.Bit 11 - File was not registered in the system registry.
StopVerify Set this parameter to True to stop the verification process.
File verification is initiated using the VerifyMode or VerifyFile properties, and the VerifyObjectFile method.

You can also use the CF_ constants as masks to determine the value of a particular bit. These constants are defined in file vervrfy.bas.

FileScan(ReferenceFile As String, VerifyFlags As Long, StopScan As Integer)

VersionStamper allows you to embed version information and warning conditions for components into your Visual Basic executable. During file scanning, the VersionStamper control scans through this embedded information and triggers a FileScan event for each file found using the following parameters:

Parameter **Description**
ReferenceFile The embedded file name. This will include a full or relative path only if specified during the embedding process.
VerifyFlags A Long that determines the warning conditions for this file. It can be a combination of the following:Bit 0 - Date & Time is specified.Bit 1 - Version number is specified.Bit 2 - Use full path (no search).Bit 3 - Use relative path (no search).Bit 16 - Trigger warning on older file.Bit 17 - Trigger warning on newer file.Bit 18 - Trigger warning if special flags set.Bit 19 - Base warning on version comparison.Bit 20 - Base warning on date comparison.Bit 21 - Always trigger a warning for this file.Bit 22 - Trigger warning on larger file size.Bit 23 - Trigger warning on smaller file size.Bit 24 - Trigger warning if file is not registered in system registry.
StopScan Set this parameter to True to stop the verification process.
File verification is started using the VerifyMode or the ScanFile properties.

You can also use the VF_ constants as masks to determine the value of a particular bit. These constants are defined in file verenum.bas.

EnumComplete()

This event is triggered after file scanning or verification to indicate that the operation is complete. This event has no parameters.

Starting the Verification or Scan

The VerifyMode and VerifyFile properties control when VersionStamper performs the verification.

VerifyMode
This property is used to control verification of the component information embedded into your Visual Basic application. Each of the embedded components is verified against the components specified in the VersionStamper control for which the property is set.

There are four possible values for this property:

Value **Description**
0 None - Verification is not triggered.
1 On Load - Verification is triggered as soon as the form containing the VersionStamper custom control is completely loaded.
2 Manual Verify - Setting the VerifyMode property to this value causes an immediate component verification operation to take place. Any files on the system that trigger a warning condition according to the embedded component list will cause a FileConflict event to be triggered.
3 Manual Scan - Setting the VerifyMode property to this value causes the list of embedded components to be enumerated by the FileScan event for this control.
The only values that are valid at design time are 0 and 1. Values 2 and 3 can be set at runtime under program control to trigger the corresponding operation. The property's value is automatically set back to zero upon completion, so this operation can be repeated as often as desired. You cannot perform another verify or scan with the same VersionStamper control if one is already in progress.

VerifyFile

While the VerifyMode property can be used to control verification of the application containing the VersionStamper control. During this verification process, a FileConflict event is triggered for each conflict that is detected between the embedded component information and the actual components located on the target system.

The VerifyFile property allows you to perform this verification process for any file that contains embedded information. This is frequently used by rescue programs such as verresq.exe to allow you to detect conflicts for files that cannot even load.

To start a verification, simply set this property to the name of the file. You cannot perform another verify or scan with the same VersionStamper control if one is already in progress.

Embedding a component list into your executable.

Using the Select Files Dialog Box

A component list is embedded into your program using the SelectFiles property.

1. If you wish to use the automatic scanning capability to search your project for all components (OCXs, VBXs, DLLs, and other specified files) used by the project, first save your project. You should be sure that all forms and modules are saved as text rather than binary (Visual Basic version 3.0 only - this is considered good VB programming practice in general and is recommended by Microsoft as well). The automatic scanning capability is available only for Visual Basic projects.

2. Chose the SelectFiles property and press the button to bring up the Property Page, then select the Select Files button to bring up the Select Files dialog box. It will take a few seconds for the dialog to come up as VersionStamper scans your system for available components. Note: you may wish to filter the paths and files in which the control searches for available components prior to bringing up the Select Files dialog box (OCX edition only). Also, the Select Files dialog box is different in the VBX edition.

The Available list box contains a listing of all components available found in the current directory, Windows directories, system path, or other specified paths on your system. You can load or remove files from this list box using the Directory & File Filter (OCX edition only) button.

3. VersionStamper allows you to embed two lists of components (OCX edition only) to verify depending upon whether your compiled program will be a 16 or 32 - bit executable. Select the list that you want to create (the default select list will be in the same environment as the VersionStamper control).

4. If you wish to use the automatic scanning feature (to scan Visual Basic projects only) of this control, press the Scan Project button. This brings up the project scan dialog that lets you choose a Visual Basic project to scan. This feature automatically adds the current version of all Objects (in the OCX edition, VersionStamper scans the project file for the Object=keyword) used by a project into the Selected list box. VersionStamper then scans every Declare statement in your project files and adds any DLLs called into the Selected list box as well. Refer to the Using the Select Files dialog description in the Embedded File Information section for more information and limitations of this feature.

5. Add any additional components from the Available list box into the Selected list box by double clicking on their entries in the Available list box, or selecting them and pressing the Add

button.

6. When you select each entry in Selected list box, the current warning status for that file will appear in the Trigger Warnings section. The default is always to warn on older versions based upon the version stamp. Files that do not have embedded version resources will be set to warn on older versions based upon the file date and time. This default is appropriate for most applications. If a component is registered in the Registry, the Search Registry check box is automatically selected for you.

7. Click on the OK button to close the dialog box.