Welcome to the Xceed Zip Compression Library v4.0

The Xceed Zip Compression Library is a high-performance data compression library for Windows developers. It provides your applications with the capability to create and manipulate industry-standard zip files as well as compress and decompress strings and buffers directly in memory.

This 4th generation Xceed Zip Compression Library is designed to offer the ultimate in flexibility while being the easiest to use product of its kind. It allows most data compression tasks to be accomplished with only a few lines of code. Xceed Zip is ideal for zipping or unzipping files on demand, compressing database fields, preparing files for transmission, automating backups, and compressing your application's data files.

The fully self-contained ActiveX control at the core of the library is all you need to include with your projects. The component has the latest generation zip compression engine built-in, and has no external dependencies such as MFC DLLS, runtime libraries or other resources. That translates to minimal hassle at distribution time. The library also supports all the available ActiveX functionality designed to make your programming job easier, such as categorized properties, built-in constants and enumerations, context-sensitive F1-help, optional parameters and built-in help strings.

Written entirely with threadsafe code and supporting both the single-threaded and multi-threaded appartment models, this edition of the library has very low memory requirements and is perfectly suited for use in environments running multiple zip/unzip operations simultaneously. It is at home on Active Server Pages, can run in the background, and can be dynamically instantiated for formless use.

The library comes with support for most of today's 32-bit development languages, including Visual Basic, Visual C++, Delphi, C++Builder, Office 97 suite products, Visual FoxPro, Visual Objects and more. Xceed Zip provides you with carefully designed properties, methods and events that make development as close to intuitive as currently possible. For those times when you're not sure, pressing F1 on any keyword will pop-up our world class help documentation. The goal: We want you to spend the least amount of time fiddling with the compression portion of your application.

We invite you to download the fully functional free trial version of the library and start integrating Xceed Zip into one of your existing programs right away. All you really have to do is add the control to a form, set one or two properties, and off you go! Get it at http://www.xceedsoft.com.

All Xceed products include comprehensive online documentation, free expert technical support, royalty-free distribution and a 60-day money-back guarantee. If you're not satisfied for any reason, return the product for a full refund of the purchase price.

Xceed is devoted to the development and support of the Xceed Zip Compression Library and is determined to pursue its evolution. That's a promise we have kept since the library's introduction in 1995. Visit our web site at www.xceedsoft.com for up-to-date information on upgrades, add-on modules for Xceed Zip and other new Xceed developer tools.

Features of the Xceed Zip Compression Library

ActiveX technology

- A single fully self-contained COM object and ActiveX control written with ATL 3.0
- Requires no external compression DLLs, MFC DLLs or runtime libraries
- Both single-threaded (STA) and multi-threaded apartment (MTA) model design
- Instantiate the component dynamically without putting it on a form
- Supports multiple instances running simultaneously
- Supports both Unicode and Ansi in one single control. The library automatically uses Unicode API calls for best performance when running on Windows NT.
- ActiveX control interface has all constants, enumerations and types built-in. Code completion, property categories, help strings and context-sensitive F1-help fully supported
- Works with all 32-bit development environments that can use ActiveX controls, including VB, VC++, Delphi, C++ Builder, VFP, Office, PB, VO, Web scripting languages, ASP and DCOM projects

Zip compression and decompression engine

- Ultra fast, low memory usage compression engine
- 100% compatible with the existing PKZip 2.04g format
- Regular, spanned and self-extracting zip files created by Xceed Zip can be used by any of today's existing unzipping applications, and vice-versa
- Stores and retrieves the latest zip file format extensions, allowing Unicode filenames and NT file attributes, extra time stamps and security permissions to be stored in the zip file
- No need to repair corrupted zip files. Xceed Zip automatically tries to recover from errors and reports any inconsistencies found in the zip file while unzipping

Major operations

- Zip files or memory buffers into new or existing zip files
- Unzip files to disk or directly to memory
- Compress and decompress strings or buffers completely in memory
- All compression and decompression modes support streaming and encryption
- Create fully customized self-extracting zip files with the optional Self-Extractor Module
- Convert existing zip files into self-extracting zip files
- Reads and writes zip files that span multiple disks
- Reads and writes multi-part zip files directly to hard drive
- Optional background processing so your code can continue execution while Xceed Zip is working

Application-component interactivity

- All operations can be aborted at any time
- Provides status report events on a file-by-file basis as well as on the entire operation being performed, with various percentages, byte and file counts. Easily add any kind of progress bar to your application.
- Events are triggered whenever a file is being zipped, unzipped, updated, deleted, etc.
- Notifies your application when and why a specific file cannot be processed.
- Notifies your application when a file being zipped or unzipped is about to overwrite an existing file. Your application can then choose to overwrite it or skip it. You can also rename files.
- Provides complete file information for the current file being processed.
- Your application receives info such as a file's achieved compression ratio, compressed size and other info as it becomes available.
- Informs your application that it should prompt the user to insert a specific disk reading or writing zip files that span multiple disks.

Flexibility features

- Process or exclude files based on file attributes, sizes, version resources and dates – or create your own custom include/exclude filters
- File previewing - find out exactly which files will match your wildcards and filters before you start zipping or unzipping. You can also obtain the compressed file sizes before starting the actual zipping operation. Statistics on the entire group of files previewed are also provided.

Zip file manipulation functionality

- Obtain global information on a zip file
- Obtain a detailed list of a zip file's contents
- Delete files from within a zip file
- Test the integrity of a zip file and its contents
- Add and retrieve comments for individual files in the zip file
- Get and set the global zip file comment
- Automatic full yield so other applications do not halt during processing.
- Clear disks before writing the zip file to them
- Compress entire directories and their contents
- Process only files newer than those already in the zip file or existing on disk
- Provide global encryption and decryption passwords or on a file-by-file basis
- Control which types of files are stored in the zip file without compression
- Control what path information is stored in the zip file for each file
- Store a drive's volume label in the zip file and restore it during unzipping
- Zip up system and hidden files, overwrite read-only files
- Allow zipping files that are currently open for write operations by other process
- Limit operation to files with timestamps and file sizes within specified ranges
- Supports long filenames, UNC paths and stores Unicode filenames in the zip file
- Manipulate self-extracting zip files without disturbing their self-extracting ability
- Rename files and their paths as they are being zipped or unzipped
- Control the amount of compression applied to files being zipped
- Specify the path where files will be uncompressed to
- Specify files to process using wildcards
- Work on temporary copies of zip files, for maximum safety.
- Specify the location of where temp files are created

Features of the Xceed Zip Self-Extractor Module

Major operations

- Create new self-extracting zip files or transform already existing zip files into self-extracting zip files.
- Select either 16-bit self-extracting zip files that work on all Windows 3.x, Windows 95, 98 and NT operating systems, or 32-bit self-extracting zip files that support long filenames and work on Windows 95, 98 and NT.
- Create self-extracting zip files that span multiple disks.
- Customize the self-extracting zip files using properties built into the Xceed Zip Compression Library components, or use configuration files.

Customization features

- Introduction message and dialog box titles are customizable.
- Display a custom license agreement with configurable accept and refuse buttons.
- Customize all other messages, prompts and button captions too.
- Customize the self-extracting zip file's application icon.
- Customizable default unzipping folder.

User interaction

- Display a confirmation dialog that allows users to select an alternate unzipping folder.
- Display a password dialog box whenever an encrypted file is encountered. The user can enter the decryption password or skip the file if they don't have the password for the file.
- Display a text file after successfully unzipping files.
- Customizable overwrite behavior with dialog box offering the user various overwrite options.
- Display a detailed status dialog during extraction.
- Any on-screen messages and dialogs can be shut off for quiet operation.

Other features

- Create self-extracting zip files with a built-in decryption password. This prevents users from using unzipping programs to unzip the files, and therefore forces users to view your custom introduction, license agreement or warning messages.
- Create program groups and insert items into program groups.
- Associate filename extensions with applications.
- Execute an application or open a document using its associated application after successfully unzipping files. Optional command-line parameters can also be specified.
- InstallMode feature allows 3rd-party install or setup programs (and their data files) to be unzipped into a temporary folder, executed, and then deleted upon completion.
- Many customized paths, filenames, strings and prompts are parsed, allowing you to insert strings such as the current folder, windows and windows system directories, the temporary folder, and more.

What's new since version 3.5

Every aspect of the library was completely revised. Interface, compression engine, features, flexibility and this documentation. To be more precise, this is a completely different library altogether. The interface has been improved, and we have added the following new capabilities that were not available in Xceed Zip v3.5:

- Brand new zip and unzip compression engine that uses very little memory and runs faster.
- The entire library was written with multi-threading in mind, and is designed to efficiently run multiple zipping and unzipping operations simultaneously.
- The library now consist of a single, fully self-contained COM object and ActiveX control. For those of you who care about these things, it was written with ATL 3.0 in Visual C++ 6.0, fully beta tested and processed with, among other tools, Numega's BoundsChecker (for memory and resource leaks and API calling accuracy), TrueCoverage (to make sure every line of code has been tested) and TrueTime for optimization.
- Requires no external DLLs. No MFC.
- Both single threaded (STA) and multi-threaded apartment (MTA) model.
- Instantiate the control dynamically without putting it on a form.
- Supports multiple instances running simultaneously.
- Both UNICODE and ANSI supported in the same control
- Detects when running on Windows NT and uses UNICODE API calls for faster operation
- ActiveX control interface has all constants, enumerations and types built-in. Code completion, property categories, help strings and context-sensitive F1-help fully supported.
- Stores/Retrieves NT file attributes and security permissions in the zip file.
- Powerful, flexible file to memory and memory to file zipping/unzipping with streaming support.
- Memory-only compression with streaming and encryption.
- Create multi-part zip files directly to hard drive.
- Process or exclude files based on file attributes, sizes, version resource and dates. Create your own custom include/exclude filters.
- File previewing - find out exactly which files will match your wildcards before you start zipping or unzipping. Get statistics on these files too.
- Obtain compressed sizes before starting the zipping operation.
- No need to repair corrupted zip files. Xceed Zip automatically tries to recover, and reports any inconsistencies found in the zip file while unzipping.
- Automatic disk spanning. No need to set a property indicating what kind of zip file you'll be unzipping.
- Can unzip spanned zip files without first inserting the last disk.
- Works with all 32-bit development environments that can use ActiveX controls, including VB, VC++, Delphi, C++ Builder, VFP, Office, PB, VO, Web scripting languages, ASP and DCOM projects.

About the free trial version

The free trial version of the Xceed Zip Compression Library v4.0 includes everything the registered version does, except that it does not allow you to distribute applications you create that were created using the trial version of the library. That means you get the complete Windows help manual with examples, sample applications with fully commented source codes, **free technical support by email or telephone** and the Xceed Zip Self-Extractor Module.

The trial version allows you to design, code and test applications that use Xceed Zip – as long as you don't try to distribute them to other machines. Only registered users are permitted to distribute applications that use the free trial version.

When you have decided that you like the product and will use it, all you have to do after you have purchased a registered copy is to install the registered package and recompile your application(s). You could wait until the last minute before registering (because Xceed Software processes orders quickly and sends registered copies by email) but you run the risk of either forgetting, running into delays with your purchasing department or being delayed by other unforeseen problems.

Please see the How to order section for ordering information.

Zipping files

Basic zipping

To zip up files, perform the following 4 steps. Put the Xceed Zip control on a form, then:

- **Specify the zip file to create or add files to**. To do this, set the ZipFilename property.
- **Specify the files you want to zip up**. To do this, set the FilesToProcess property.
- **Tell Xceed Zip to start zipping**. To do this, call the Zip method.
- **Make sure it worked successfully**. To do this, check the Zip method's return value.

Here is sample code for these 4 steps:

```
' Don't forget to put an Xceed Zip control on a form
Dim ResultCode As xcdError
XceedZip1.ZipFilename = "c:\backups\My documents.zip"
XceedZip1.FilesToProcess = "c:\documents\word\*.doc"
ResultCode = XceedZip1.Zip
If ResultCode = xerSuccess then
    MsgBox "Files zipped successfully."
Else
    MsgBox XceedZip1.GetErrorDescription(xvtError, ResultCode)
EndIf
```

Things you should consider

The main questions you should ask yourself when zipping files are:

- **Do you want to store paths in the zip file?** See the controlling how paths are stored topic.
- **Do you want to zip up files in subfolders, too?** See the ProcessSubfolders property.
- **Do you want to display the status of the zipping operation?** See the GlobalStatus event.
- **Is there a chance the zip file may span multiple disks?** If so, write a handler for the InsertDisk event.

Other questions you may want to consider are:

- **Do you want to zip only certain types of files?** See the filtering properties.
- **Do you want to password-protect your zipped files?** See the EncryptionPassword property.
- **Do you want to edit a file's info before zipping?** See the ZipPreprocessingFile event.
- **Is there a chance you're zipping files already open?** See the ZipOpenedFiles property.
- **Are you looking for the best compression ratio?** See the CompressionLevel property.
- **Do you want to store Unicode filenames in the zip file?** See the ExtraHeaders property.

If there's anything else you need to do that's not mentioned here, consult the Properties topic, or search the index – chances are, Xceed Zip does what you need. If not, ask Xceed Software whether it is possible or not.

Unzipping files

Basic unzipping

To unzip files, perform the following 4 steps. Put the Xceed Zip control on a form, then:

- **Specify the zip file to unzip files from**. To do this, set the ZipFilename property.
- **Specify where to unzip files to**. To do this, set UnzipToFolder property.
- **Tell Xceed Zip to start unzipping**. To do this, call the Unzip method.
- **Make sure it worked successfully**. To do this, check the Unzip method's return value.

Here is sample code for these 4 steps:

```
' Don't forget to put an Xceed Zip control on a form
Dim ResultCode As xcdError
XceedZip1.ZipFilename = "c:\backups\My documents.zip"
XceedZip1.UnzipToFolder = "c:\documents\word"
ResultCode = XceedZip1.Unzip
If ResultCode = xerSuccess then
    MsgBox "Files unzipped successfully."
Else
    MsgBox XceedZip1.GetErrorDescription(xvtError, ResultCode)
EndIf
```

Things you should consider

The main questions you should ask yourself when unzipping files are:

- **Do you want to unzip only some files, not all of them?** See the FilesToProcess property.
- **Do you want to recreate paths stored in the zip file?** See the PreservePaths property.
- **Do you want to display the status of the unzipping operation?** See the GlobalStatus event.
- **Are you possibly unzipping a spanned zip file?** If so, write a handler for the InsertDisk event.

Other questions you may want to consider are:

- **Do you want to unzip only certain types of files?** See the filtering properties.
- **Are the files you are unzipping password-protected?** See the EncryptionPassword property.
- **Do you want to edit a file's info before unzipping?** See the UnzipPreprocessingFile property.

If there's anything else you need to do that's not mentioned here, consult the Properties topic, or search the index – chances are, Xceed Zip does what you need. If not, ask Xceed Software whether it is possible or not.

Compression entirely in memory

Compression entirely in memory

Compression entirely in memory

To compress entirely in memory, from one memory buffer to another, use the Xceed Compression control's Compress method. Keep in mind that this is a different control than the Xceed Zip control. To uncompress data that has been compressed this way, use the Xceed Compression control's Uncompress method. The output of the Compress method is always a byte array variant.

Here is an example for VB users:

```
' Don't forget to put an Xceed Compression control on a form
Dim CompressedData As Variant
Dim OriginalData As Variant
XceedCompression1.Compress "This is some text to compress", CompressedData,
    True
XceedCompression1.Uncompress CompressedData, OriginalData, True
Debug.Print OriginalData
```

In the example above, the return values of the Compress and Uncompress methods were ignored. Also, an encryption password was not used. However, you could add error checking and passwords, as in the following example:

```
Dim CompressedData As Variant
Dim OriginalData As Variant
Dim ResultCode As xcdError
XceedCompression1.EncryptionPassword = "the password"
ResultCode = XceedCompression1.Compress("This is some text to compress",
    CompressedData, True)
If ResultCode <> xceSuccess Then MsgBox "Error #" + Str(ResultCode) +
    " while compressing the data."
ResultCode = XceedCompression1.Uncompress(CompressedData, OriginalData, True)
If ResultCode <> xceSuccess Then MsgBox "Error #" + Str(ResultCode) +
    " while decompressing the data."
Debug.Print OriginalData
```

Listing the contents of a zip file

Basic listing

To list the contents of a zip file, perform the following 4 steps. Put the Xceed Zip control on a form, add a listbox to the form if that's where you want to list files in, and then:

- **Specify the zip file to list the contents of**. To do this, set the ZipFilename property.
- **Tell Xceed Zip to start listing**. To do this, call the ListZipContents method.
- **Obtain each filename and file info**. To do this, write a handler for the ListingFile event.
- **Make sure it worked successfully**. To do this, check the ListZipContent method's return value.

Here is sample code for these 4 steps (excluding the code for the ListingFile event):

```
' Don't forget to put an Xceed Zip control on a form
Dim ResultCode As xcdError
XceedZip1.ZipFilename = "c:\backups\My documents.zip"
ResultCode = XceedZip1.ListZipContents
If ResultCode = xerSuccess then
    MsgBox "Files listed successfully."
Else
    MsgBox XceedZip1.GetErrorDescription(xvtError, ResultCode)
EndIf
```

Here is sample code to put into the ListingFile event handler. This sample assumes you want to put the listed items in a listbox, and display only the filename and size:

```
ListBox1.AddItem sFilename + " size=" + Str(lCompressedSize)
```

Things you should consider

The main questions you should ask yourself when listing files are:

- **What information do you need?** The ListingFile event provides plenty of file information.
- **Are you possibly listing a spanned zip file?** If so, write a handler for the InsertDisk event.
- **Do you want to display the status of the listing operation?** See the GlobalStatus event.

Other questions you may want to consider are:

- **Do you want to list only certain types of files?** See the filtering properties.

If there's anything else you need to do that's not mentioned here, consult the Properties topic, or search the index – chances are, Xceed Zip does what you need. If not, ask Xceed Software whether it is possible or not.

Installation instructions for Visual Basic 5.0

The Xceed Zip Compression Library setup utility will copy all the necessary files to your system and place the XCEEDZIP.DLL in the Windows System folder and register it with the operating system. The only thing left to do is to make the control available to your VB 5.0 projects. To do this, perform the following steps:

- Start Visual Basic 5.0.
- Create a new project or open an existing one.
- Select Components… from the Project menu.
- An entry named "Xceed Zip Compression Library v4.0" entry should appear in the components list.
- In the Components dialog, make sure the "Xceed Zip Compression Library v4.0" entry is checked in the components list. This will make it available for use in your applications.
- Click the OK button. The Xceed Zip and Xceed Compression control icons should now appear in the components toolbox.

If the control is not in the components list, perform the following steps:

- Try reinstalling the Xceed Zip Compression Library from its setup program. Continue only if that didn't help.
- Click the Browse… button.
- Find and select the XCEEDZIP.DLL file (use the one in the Windows System folder that was copied there by the setup program) then click on the OK button.
- The "Xceed Zip Compression Library v4.0" entry should now appear in the components list.

Installation instructions for Visual Basic 6.0

The Xceed Zip Compression Library setup utility will copy all the necessary files to your system and place the XCEEDZIP.DLL in the Windows System folder and register it with the operating system. The only thing left to do is to make the control available to your VB 6.0 projects. To do this, perform the following steps:

- Start Visual Basic 6.0.
- Create a new project or open an existing one.
- Select Components… from the Project menu.
- An entry named "Xceed Zip Compression Library v4.0" entry should appear in the components list.
- In the Components dialog, make sure the "Xceed Zip Compression Library v4.0" entry is checked in the components list. This will make it available for use in your applications.
- Click the OK button. The Xceed Zip and Xceed Compression control icons should now appear in the components toolbox.

If the control is not in the components list, perform the following steps:

- Try reinstalling the Xceed Zip Compression Library from its setup program. Continue only if that didn't help.
- Click the Browse… button.
- Click on All files in the Files of type tab
- Find and select the XCEEDZIP.DLL file (use the one in the Windows System folder that was copied there by the setup program) then click on the OK button.

The "Xceed Zip Compression Library v4.0" entry should now appear in the components list.

Installation instructions for Visual C++ 6.0 (MFC projects)

The Xceed Zip Compression Library setup utility will copy all the necessary files to your system and place the XCEEDZIP.DLL in the Windows System folder and register it with the operating system. The only thing left to do is to make the control available to your VC++ 6.0 projects. To do this, perform the following steps:

- Start Microsoft Visual C++ 6.0
- Open your MFC AppWizard application project, or create a new one by selecting the New... menu item from the File menu, then selecting the Projects tab and choosing the MFC AppWizard item. Follow the Wizards instructions to create the project. Be sure that you select the ActiveX Controls option for the question "What other support would you like to include?"
- Select the Project menu
- Select the Add to project submenu, then select Components and Controls...
- In the Components and Controls Gallery dialog, look in the Registered ActiveX Controls folder for the "Xceed Zip Control 4.0" or the "Xceed Compression control" entry, then select it and click on the Insert button.
- Select the OK in response to the Insert this component? message box. The Confirm Classes dialog should appear.
- Click OK to close the Confirm Classes dialog.
- Click the Close button to close the Components and Controls Gallery dialog.
- The XCEEDZIP.CPP and XCEEDZIP.H files should now be part of your project. You will now be able to instantiate the Xceed Zip object in your project, and see the Xceed Zip icon available in your Controls toolbar when designing dialog boxes.

Installation instructions for Word 97 and Excel 97

The Xceed Zip Compression Library setup utility will copy all the necessary files to your system and place the XCEEDZIP.DLL in the Windows System folder and register it with the operating system. The only thing left to do is to make the control available to your Word 97 projects. To do this, perform the following steps:

- Start Word 97 or Excel 97.
- Select the Visual Basic Editor option from the Macro submenu in the Tools menu or use the Alt+F11 shortcut.
- In the Microsoft Visual Basic window, select the References… option from the Tools Menu.
- Make sure the "Xceed Zip Compression Library v4.0" entry is checked and then click on the OK button.
- In the Microsoft Visual Basic window, select the UserForm option from the Insert menu.
- If the Toolbox window is not visible, select the ToolBox option from the View menu.
- Select the Additional Controls… option from the Tools menu.
- In the Additional Controls dialog, make sure "Xceed Zip Compression Library v4.0" is checked in the Available Controls list.
- Click the OK button.

If there is no entry for Xceed Zip in the references dialog's list of available references:

- Try reinstalling the Xceed Zip Compression Library from its setup program. Continue only if that didn't help.
- Try reinstalling the Xceed Zip Compression Library from its setup program. Continue only if that didn't help.
- In the References dialog, click the Browse... button.
- Find and select the XCEEDZIP.DLL file (preferably the one in Windows System which was placed there by the Xceed Zip setup program), then click on the Open button.
- The "Xceed Zip Compression Library v4.0" entry should now appear in the Available References listing in the References dialog.

Installation instructions for Visual FoxPro 5.0 and 6.0

The Xceed Zip Compression Library setup utility will copy all the necessary files to your system and place the XCEEDZIP.DLL in the Windows System folder and register it with the operating system. The only thing left to do is to make the control available to your VFP 5.0 and 6.0 projects. To do this, perform the following steps:

- Start Visual FoxPro 5.0 or 6.0
- Select Options... from the Tools menu.
- Select the Controls tab from the Options dialog.
- Select the ActiveX controls checkbox.
- The Xceed Zip control should appear in the "selected:" listbox (still in the Controls tab of the Options dialog)
- Now, to make the control available to your Visual FoxPro 5.0 projects, make sure the checkbox next to the "Xceed Zip Compression Library v4.0" entry is selected. Click on the OK button and the Xceed Zip icon should now appear in the list of controls you can insert - for example, when using the OLE Container Object on a form.

If the entry for Xceed Zip does not appear in the list of controls:

- Try reinstalling the Xceed Zip Compression Library from its setup program. Continue only if that didn't help.
- Select the Add... button from the Controls tab of the Options dialog.
- Find and select the XCEEDZIP.DLL file (preferably the one in the Windows System folder that was placed there by the Xceed Zip setup program), then click on the OK button.
- The control's entry should now appear in the list.

Special note for Visual FoxPro 5.0 and 6.0: Xceed Zip events will be triggered only if you set "Application.AutoYield = .F." Place this code in the section of your main program where you set up your environment. If you only have a single form in your application that uses Xceed Zip, place "Application.AutoYield = .F." in the forms load method, and set it back to the default in the forms destroy method.

Installation instructions for Access 97

The Xceed Zip Compression Library setup utility will copy all the necessary files to your system and place the XCEEDZIP.DLL in the Windows System folder and register it with the operating system. The only thing left to do is to make the control available to your VB 4.0 projects. To do this, perform the following steps:

- Start Access 97.
- Select the ActiveX Controls… option from the Tools menu
- The "Xceed Zip Compression Library v4.0" entry should appear in the Available Controls list of the ActiveX Controls dialog.
- Click the Close button.

If the Xceed Zip entry does not appear in the list:

- Try reinstalling the Xceed Zip Compression Library from its setup program. Continue only if that didn't help.
- In the ActiveX Controls dialog, click the Register... button.
- Find and select the XCEEDZIP.DLL file (use the one in the Windows System folder that was copied there by the setup program) then click on the Open button.
- The Xceed Zip control should now be available the "Available Controls" list.

Installation instructions for Delphi 3.0

The Xceed Zip Compression Library setup utility will copy all the necessary files to your system and place the XCEEDZIP.DLL in the Windows System folder and register it with the operating system. The only thing left to do is to make the control available in your Delphi 3.0 environment by inserting the control into the toolbar. To do this, perform the following steps:

- Start Delphi 3.0
- Open an existing project or create a new project
- From the Component menu, select the Import ActiveX control item
- Select the "Xceed Zip Compression Library 4.0" item from the list of installed ActiveX controls (if the component is not there, we recommend that you reinstall the library on your machine instead of choosing the Add button and selecting the XCEEDZIP.DLL control from the Windows System directory)
- Click on the Install button that's on the Import ActiveX dialog
- The Install dialog will appear. Click on the "Into new package" tab.
- Enter a filename and path for the new package in the "File Name" field. It should be a ".DPK" file, and you'll probably want to put it with your other package files in your Delphi installation folder.
- Click OK in the Install dialog
- Delphi will then ask you to confirm the rebuild – click Yes in the Confirm dialog.
- If everything was successful, you should get a message that the package has been updated and that the new components have been registered.
- Close the package dialog
- Click Yes when you are prompted to confirm the changes.

You have now succesfully installed the Xceed Zip Compression Library v4.0 ActiveX control in Delphi 3. The components should appear in your component toolbar (ActiveX or Xceed tabs).

Installation instructions for Delphi 4.0

The Xceed Zip Compression Library setup utility will copy all the necessary files to your system and place the XCEEDZIP.DLL in the Windows System folder and register it with the operating system. The only thing left to do is to make the control available in your Delphi 4.0 environment by inserting the control into the toolbar. To do this, perform the following steps:

- Start Delphi 4.0
- Open an existing project or create a new project
- From the Component menu, select the Import ActiveX control item
- Select the "Xceed Zip Compression Library 4.0" item from the list of installed ActiveX controls (if the component is not there, we recommend that you reinstall the library on your machine instead of choosing the Add button and selecting the XCEEDZIP.DLL control from the Windows System directory)
- Click on the Install button that's on the Import ActiveX dialog
- The Install dialog will appear. Click on the "Into new package" tab.
- Enter a filename and path for the new package in the "File Name" field. It should be a ".DPK" file, and you'll probably want to put it with your other package files in your Delphi installation folder.
- Click OK in the Install dialog
- Delphi will then ask you to confirm the rebuild – click Yes in the Confirm dialog.
- If everything was successful, you should get a message that the package has been updated and that the new components have been registered.
- Close the package dialog
- Click Yes when you are prompted to confirm the changes.

You have now succesfully installed the Xceed Zip Compression Library v4.0 ActiveX control in Delphi 4. The components should appear in your component toolbar (ActiveX or Xceed tabs).

Installation instructions for C++ Builder 3.0

The Xceed Zip Compression Library setup utility will copy all the necessary files to your system and place the XCEEDZIP.DLL in the Windows System folder and register it with the operating system. The only thing left to do is to make the control available in your C++ Builder 3.0 environment by inserting the control into the toolbar. To do this, perform the following steps:

- Start C++ Builder 3.0
- Open an existing project or create a new project
- From the Component menu, select the Import ActiveX control item
- Select the "Xceed Zip Compression Library 4.0" item from the list of installed ActiveX controls (if the component is not there, we recommend that you reinstall the library on your machine instead of choosing the Add button and selecting the XCEEDZIP.DLL control from the Windows System directory)
- Click on the Install button that's on the Import ActiveX dialog
- The Install dialog will appear. Click on the "Into new package" tab.
- Enter a filename and path for the new package in the "File Name" field. It should be a ".BPK" file, and you'll probably want to put it with your other package files in your C++ Builder installation folder
- Click OK in the install dialog
- If everything was successful, you should get a message that the package has been updated and that the new components have been registered.
- Close the package dialog
- Click Yes when you are prompted to confirm the changes.

The components should now appear in the ActiveX tab of your component toolbar.

Installation instructions for C++ Builder 4.0

The Xceed Zip Compression Library setup utility will copy all the necessary files to your system and place the XCEEDZIP.DLL in the Windows System folder and register it with the operating system. The only thing left to do is to make the control available in your C++ Builder 4.0 environment by inserting the control into the toolbar. To do this, perform the following steps:

- Start C++ Builder 4.0
- Open an existing project or create a new project
- From the Component menu, select the Import ActiveX control item
- Select the "Xceed Zip Compression Library 4.0" item from the list of installed ActiveX controls (if the component is not there, we recommend that you reinstall the library on your machine instead of choosing the Add button and selecting the XCEEDZIP.DLL control from the Windows System directory)
- Click on the Install button that's on the Import ActiveX dialog
- The Install dialog will appear. Click on the "Into new package" tab.
- Enter a filename and path for the new package in the "File Name" field. It should be a ".BPK" file, and you'll probably want to put it with your other package files in your C++ Builder installation folder
- Click OK in the install dialog
- If everything was successful, you should get a message that the package has been updated and that the new components have been registered.
- Close the package dialog
- Click Yes when you are prompted to confirm the changes.

The components should now appear in the ActiveX tab of your component toolbar.

Installation instructions for other 32-bit development environments

The Xceed Zip ActiveX control should be available in your languages list of available ActiveX controls. Normally, the component (XCEEDZIP.DLL) is copied to the Windows System folder by the Xceed Zip setup program and then "registered" with the operating system. Your language should automatically "see" the component. If it does not, then you may need to manually register the control. Before doing this, though, we strongly suggest that you reinstall your Xceed Zip Compression Library. This will ensure that it is licensed as well as being registered with the operating system. If that doesn't solve the problem, here's how to manually register the control:

- From the Run option in the Windows 95 or Windows NT start menu, or from a command prompt, run the following command to register an OCX:

  ```
  regsvr32 "path and filename of the xceedzip.dll"
  ```

- You will then see a message box that either displays "LoadLibrary failed" or displays "LoadLibrary succeeded". If you get the success message, then your ActiveX control is now registered on the machine and is ready for use by applications or in your development environment.

Creating a new self-extracting zip file

Creating a self-extracting zip file directly from a group of files

The method for creating a self-extracting zip file is very similar to that of creating a regular zip file, except that you must specify the self-extractor binary to use. You'll also want to (although it is optional) give your zip file a ".EXE" extention instead of the usual ".ZIP".

Here we'll follow the same steps as the zipping files topic, but modified for creating a self-extracting zip file. To create a new self-extracing zip file, perform the following 5 steps. Put the Xceed Zip control on a form, then:

- **Specify the zip file to create. Make it a ".EXE"**. To do this, set the ZipFilename property.
- **Specify the files you want to zip up**. To do this, set the FilesToProcess property.
- **Specify the self-extractor binary to use.** To do this, set the SfxBinaryModule property.
- **Tell Xceed Zip to start zipping**. To do this, call the Zip method.
- **Make sure it worked successfully**. To do this, check the Zip method's return value.

Here is sample code for these 5 steps:

```
' Don't forget to put an Xceed Zip control on a form
Dim ResultCode As xcdError
XceedZip1.ZipFilename = "c:\test\A self-extracting zip file.exe"
XceedZip1.FilesToProcess = "c:\documents\word\*.doc"
XceedZip1.SfxBinaryModule = "xcdsfx32.bin"
ResultCode = XceedZip1.Zip
If ResultCode = xerSuccess Then
    MsgBox "Created successfully."
Else
    MsgBox XceedZip1.GetErrorDescription(xvtError, ResultCode)
EndIf
```

Things you should consider

You will probably want to customize the self-extracing zip files you create. For example, you may want to change its title and introduction message, or have it suggest a default unzipping folder. There is a group of properties that start with "Sfx" designed to let you do just that. See the About the Self-Extractor Module properties topic to find out what you can customize.

Other than customizing the appearance and behavior of the self-extracting zip file, you should also ask yourself the following questions:

- **The self-extracting zip file always recreates files with their stored paths. Do you want to store paths in the zip file?** See the controlling how paths are stored topic.
- **Do you want to password-protect the files in your self-extracting zip files?** See the EncryptionPassword property. Self-extracting zip files that contain password protected files will display a dialog box asking the user to enter a decryption password, otherwise the encrypted files will be skipped. There's also the SfxDefaultPassword property that lets the self-extracting zip file decrypt some or all of its contents without asking the user for a password.
- **Do you want to zip up files in subfolders, too?** See the ProcessSubfolders property.
- **Do you want to display the status of the zipping operation?** See the GlobalStatus event.
- **Is there a chance the self-extracting zip file may span multiple disks?** If so, write a handler for the

InsertDisk event.

Other questions you may want to consider are:

- **Do you want to zip only certain types of files?** See the filtering properties.
- **Do you want to edit a file's info before zipping?** See the ZipPreprocessingFile event.
- **Is there a chance you're zipping files already open?** See the ZipOpenedFiles property.
- **Are you looking for the best compression ratio?** See the CompressionLevel property.
- **Do you want to store Unicode filenames in the zip file?** See the ExtraHeaders property.

If there's anything else you need to do that's not mentioned here, consult the Properties topic, or search the index – chances are, Xceed Zip does what you need. If not, ask Xceed Software whether it's possible or not.

Overview of the Xceed Zip control

The Xceed Zip control handles all zip and unzip tasks other than compression directly from one memory block to another. Memory-only compression is done through the Xceed Compression control. Both of these controls are part of the same component (XCEEDZIP.DLL).

The Xceed Zip control is a very flexible compression and decompression component that is fully compatible with the latest zip file format. Regular and multi-disk zip files created by Xceed Zip can be used by PKZip 2.04g, and vice-versa. Xceed Zip does not unzip files from zip files created with old versions of PKZip before PKZip 2.0 - those files must be converted to the new format.

Xceed Zip provides a large number of properties, methods and events allowing full-featured manipulation of zip files, real-time progress reports and easy integration with a graphical user interface.

Resources

- Methods
- Properties
- Events
- Enumeration types
- Error codes
- Warnings
- Skipping reasons

Getting started quickly

- Zipping files
- Unzipping files
- Listing the contents of a zip file
- Creating or unzipping a spanned zip file
- Adding a progress bar
- Creating a new self-extracting zip file
- Converting a zip file to self-extracting
- Zipping from memory to a zip file
- Unzipping from a zip file to memory
- Compression entirely in memory

Other information

- Brief introduction to the zip file format
- Controlling how paths are stored in the zip file
- Formless use of the control
- Running multiple operations at the same time

Overview of the Xceed Compression control

The Xceed Compression control handles compression and decompression when the source and destination are blocks of memory. The data can be provided in a single block of memory, or in streaming fashion. Encruption and configurable compression factors are also supported. The Xceed Compression control does not work with zip files at all. All zip and unzip tasks involving files and are handled by the Xceed Zip control. Both the Xceed Compression and Xceed Zip controls are part of the same library file (XCEEDZIP.DLL).

The Xceed Zip compression control uses the same compression algorithm that is used for zipping to and from zip files, but does not include the same type of information in the compressed data. When you compress data with the Xceed Compression control, the data does not have any file or date information embedded into it, nor does it contain local headers, a central directory, and other zip format related structures. The compressed data is almost pure raw compressed data. The only overhead added to the data is an encryption header (when applicable) and a 32-bit checksum that is used when uncompressing the data to ensure there are no errors.

Resources

- Methods
- Properties
- Enumeration types
- Error codes

Getting started quickly

- Compression entirely in memory

Overview of the Self-Extractor Module

Welcome to the Xceed Zip Self-Extractor Module!

The Xceed Zip Self-Extractor Module is an optional module that can be purchased for use with the Xceed Zip Compression Library. The Xceed Zip Self-Extractor Module provides support for creating fully customizable 16 and 32-bit self-extracting Windows zip files. These self-extracting zip files allow a user to simply "run" the zip file to unzip its contents. The self-extracting zip files contain all the code necessary to unzip their contents, so no unzipping programs, DLLs or other components are required on the user's machine in order to unzip the contents of the Zip file.

It works together with the Xceed Zip Compression Library

The Xceed Zip Compression Library lets your Windows applications transform an already existing zip file into a self-extracting zip file, or create new self-extracting zip files. Combine these capabilities with the self-extractor binary provided by the Xceed Zip Self-Extractor Module, and your Windows applications will be able to fully configure the self-extracting zip files they create. Your applications can customize the self-extractor's title, introduction message, icon, default unzip folder, text and button captions, and more. You can make it display a readme file and license agreement, run a program after the unzipping is complete, and even prompt the user for passwords to decrypt files. See the features topic for a complete list of the capabilities that the Self-Extractor Module provides.

The Xceed Zip Self-Extractor Module lets you make 16-bit self-extracting zip files that will work on all Windows 3.x, Windows 95, 98 and NT operating systems, or you can make 32-bit self-extracting zip files that support long filenames and will work on Windows 95, 98 and NT operating systems.

The properties used to configure the self-extractor are built into the Xceed Zip Compression Library components. Users who have purchased a copy of the Xceed Zip Self-Extractor Module are able to set these extra properties and start making self-extracting zip files as easily as they create regular zip files. Users who don't own a copy of the Xceed Zip Self-Extractor Module can still use the properties related to the Xceed Zip Self-Extractor Module, but in "free trial version" mode. When in free trial version mode, any self-extracting zip files created with the Xceed binaries will be considered trial versions and will inform you of this when they are executed. Feel free to try it out.

Resources

- Methods
- Properties
- Enumeration types

Getting started quickly

- Creating a new self-extracting zip file
- Converting a zip file into a self-extracting zip file

About the ActiveX component

The XCEEDZIP.DLL is a fully self-contained COM ActiveX control written entirely with ATL v3.0 and Visual C++ 6.0. There are no external dependencies required for its operation such as other DLLs.

The control's DLL file can be found in the "C:\PROGRAM FILES\XCEED ZIP\BIN" directory, if you installed your Xceed Zip package in the default location.

The ActiveX control supports both the single-threaded apartment and the multi-threaded apartment models, and therefore will not work with Windows NT 3.51 because the DCOM extensions are not available for that operating system.

The control is digitally signed with Microsoft Authenticode technology, with VeriSign as the certificate verification authority. This ensures that the component has not been tampered or modified by parties other than Xceed Software, and is also useful when the component is used on web pages so that browsers do not complain about security issues.

Typically, applications that use ActiveX controls are distributed with an install or setup program that will also take care of registering the controls on the machine. If you aren't using a setup program, then you will have to manually register the component on the machine.

How to manually register the ActiveX component

The registration process referred to here is not related to licensing, it refers to the process of making the control known to the operating system. If you don't use an install program with this capability built-in, you can register your controls manually. From the Run option in the Windows 95 or Windows NT start menu, or from a command prompt, run the following command to register an ActiveX control:

```
regsvr32 "path and filename of the ActiveX control's DLL file"
```

You will then see a message box that either displays "LoadLibrary failed" or displays "LoadLibrary succeeded". If you get the success message, then the control is now registered on the machine and is ready for use by applications or in your development environment. If you get the failure message, make sure you specified a correct path and filename, and placed it in quotes if it has spaces in it.

Registration problems

If you are unable to register a fully-self contained ActiveX control on a target machine, even after following the instructions above, Xceed is currently aware of only one possible problem that may prevent proper registration of ActiveX controls. The problem applies to systems running the original release of Windows 95 or OSR1. If these machines don't have Internet Explorer v4.0 or up, then you will need to install DCOM 95 (available from Microsoft's web site) which will update the COM support for these machines.

About the Self-Extractor binaries

The Xceed Zip Self-Extractor Module provides 16 and 32-bit Windows self-extractor binaries that the Xceed Zip control uses when creating customizable self-extracting zip files.

The binaries can be found in the "C:\PROGRAM FILES\XCEED ZIP\SFX" directory, if you installed your Xceed Zip package in the default location.

The binaries contain all the code necessary to unzip files and display their user interface. No external DLLs, Xceed Zip ActiveX control, support DLLs, components or applications are required to run self-extracting zip files that use the Xceed Zip Self-Extractor Module binaries.

The binaries are incorporated into the zip file at the moment of its creation, or after it has been created (for example, when using the Convert method). Therefore, these binaries must be distributed with your application, because they need to be found by the Xceed Zip Compression Library in order to make the Zip file self-extracting. The SfxBinaryModule property is used to provide their location to Xceed Zip.

The **XCDSFX32.BIN** file is the 32-bit binary, which works on all Windows 95/98 and Windows NT operating systems running on Intel 386-compatible CPUs. It supports long filenames, multiple-disk spanning, encryption passwords for individual files, displaying text files, running another application, and more. (See the features list for details)

The **XCDSFX16.BIN** file is the 16-bit binary, which works on 16-bit Windows 3.x as well as Windows 95/98 and Windows NT. It is more compact than the 32-bit binary, but does not support long filenames and uses a custom, non-Windows 95 style directory browser.

What must be distributed

You only need to distribute the XCEEDZIP.DLL file with your applications in order for the Xceed Zip Compression Library to work. If you are creating self-extracting zip files with the Self-Extractor Module, you'll also need to distribute the XCDSFX16.BIN or XCDSFX32.BIN file, depending on which one your application uses.

It is highly recommended to place the XCEEDZIP.DLL file in the Windows System folder, with an application that does version checking, so that an already existing version of the file is not overwritten by an older version. Xceed Software will ensure that as long as the file keeps the XCEEDZIP.DLL name, it will always be backwards compatible with older versions. So applications compiled to work with Xceed Zip v4.0 will work with future versions of this file as well.

The XCEEDZIP.DLL file is fully self-contained, which means that target machines do not need any other files (such as THE MFC40.DLL or MSVCRT40.DLL for example) in order for the XCEEDZIP.DLL to register and work on the machine. The early releases of Windows 95 (up to OSR1) require that you install the DCOM update or Internet Explorer v4.0, so that you can use apartment threaded components.

Only registered users can deploy applications that use the Xceed Zip Compression Library.

Getting Started Sample Application For VB5 and VB6

The Getting Started Sample Application for VB5 and VB6 can be found in the C:\PROGRAM FILES\XCEED ZIP\SAMPLES\GETTING STARTED\VB5 folder (or the VB6 folder).

To use the sample, load the STARTED.VBP project file, then run the program. The application itself explains everything to you on screen and with the help of pop-up tooltip boxes when you move the mouse over the controls in run mode.

Zip Manager Sample Application for VB5 and VB6

The Zip Manager Sample Application for VB5 and VB6 can be found in the C:\PROGRAM FILES\XCEED ZIP\SAMPLES\GETTING STARTED\VB5 folder (or the VB6 folder).

The sample includes fully commented source code which should complement the examples available in the online help, but in the context of a real application, not specific examples designed exclusively to demonstrate each method. We propose that you dive right into the sample's source code and take a look.

To use the sample, load the ZIPMANAGER.VBP project file, then run the program.

The Zip Manager Sample Application is a Windows program that allows you to manipulate zip files. By looking at the source code, you will find out how to zip, unzip, delete files, update and test files in a zip file using the Xceed Zip Compression Library v4.0 control. You will also see how event handlers are written for different events

The program works in a similar fashion to WinZip™. To create a zip file, you select the "New zip file" item from the File menu. A dialog box will be shown to allow you to enter a path and filename for the zip file, and then the "Add or update files" dialog box will be shown to allow you to add files to the zip file immediately. Just select the files to add, and click the "Zip!" button. Once the files have been added, the listbox will display the (new) contents of the zip file. The other options work in a similar fashion.

The status bar below the main listbox indicates what is currently happening during any zip file operation. For example, when zipping files, it will display the name of each file as it is being zipped. To the right of the status bar is a progress bar that shows you the progress of the current operation. When you are zipping tens of thousands of files, it might take a while before the progress bar starts to show the progress, because the library is scanning the drive to optain the total number of files you will be zipping – and this can take a few seconds at best.

The source code for this sample application is fully commented. Registered users of the Xceed Zip Compression Library can use this sample application, or portions of it, in their own commercial or private applications, royalty-free.

License, copyright and disclaimer

The use of this package indicates your understanding and acceptance of the following terms and conditions. This license shall supersede any verbal, or prior verbal or written, statement or agreement to the contrary. If you do not understand or accept these terms, or your local regulations prohibit "after sale" license agreements or limited disclaimers, you must cease and desist using this product immediately.

Copyright: This product is Copyright ©1995-1999 Xceed Software Inc., all rights reserved. This product is protected by Canadian and United States copyright laws, international treaties and all other applicable national or international laws. This software product and documentation may not, in whole or in part, be copied, photocopied, translated, or reduced to any electronic medium or machine readable form, without prior consent in writing, from Xceed Software Inc. and according to all applicable laws. The sole owner of this product is Xceed Software Inc.

Liability disclaimer: This product and/or license is provided as is, without any representation or warranty of any kind, either express or implied, including without limitation any representations or endorsements regarding the use of, the results of, or performance of the product, its appropriateness, accuracy, reliability, or correctness. The entire risk as to the use of this product is assumed by the user and/or licensee. Xceed Software Inc. does not assume liability for the use of this program beyond the original purchase price of the software. In no event will Xceed Software Inc. be liable for additional direct or indirect damages including any lost profits, lost savings, or other incidental or consequential damages arising from any defects, or the use or inability to use these programs, even if Xceed Software Inc. has been advised of the possibility of such damages.

Restrictions: You may not use, copy, modify, translate, or transfer the programs, documentation, or any copy except as expressly defined in this agreement. You may not attempt to unlock or bypass any "copy-protection" or authentication algorithm utilized by the program. You may not remove or modify any copyright notice, nor the About dialog or the method by which it may be invoked.

Operating license: You have the non-exclusive right to use any enclosed program only by a single person, on a single computer at a time. You may physically transfer the program from one computer to another, provided that the program is used only by a single person, on a single computer at a time. In group projects where multiple persons will use the program, you must purchase an individual license for each member of the group. Use over a "local area network" (within the same locale) is permitted provided that the program is used only by a single person, on a single computer at a time. Use over a "wide area network" (outside the same locale) is strictly prohibited under any and all circumstances.

Linking and distribution license: In order to link and distribute compiled Xceed Zip Compression Library code, including the XCEEDZIP.DLL, XCDSFX16.BIN and XCDSFX32.BIN files, in accordance to this license, you must be registered with Xceed Software Inc. as an authorized licensee. You may not reproduce or distribute copies of the complete package, any of the source, interface, code units or any license (.LIC) files, the LICENSE.TXT file, your private license key, any of the documentation, nor may you supply any means by which your users could create, modify, or incorporate the Xceed Zip Compression Library or Xceed Zip Self-Extractor Module in their own products. Violations will be prosecuted to the maximum extent possible under the law.

Back-up and transfer: You may make one copy of the program solely for "back-up" purposes, as prescribed by Canadian, United States, and international copyright laws. You must reproduce and include the copyright notice on the back-up copy. You may transfer the product to another party only if the other party agrees to the terms and conditions of this agreement, and completes and returns registration information (name, address, etc.) to Xceed Software Inc. within 30 days of the transfer. If you transfer the program you must at the same time transfer the documentation and back-up copy, or transfer the documentation and destroy the back-up copy. You may not retain any portion of the program, in any form, under any circumstance.

Terms: This license is effective until terminated. You may terminate it by destroying the program, the documentation and copies thereof. This license will also terminate if you fail to comply with any terms or conditions of this agreement. You agree upon such termination to destroy all copies of the program and of the documentation, or return them to Xceed Software Inc. for disposal.

Other rights and restrictions: All other rights and restrictions not specifically granted in this license are reserved by Xceed Software Inc.

Acknowledgments

Numega

Xceed Zip v4.0 was extensively checked and processed with the help of Numega's BoundsChecker, TrueCoverage, TrueTime software and of course, lots of beta testing! BoundsChecker finds memory and resource leaks, checks API calls for accuracy, and finds a host of other potential problems. TrueCoverage makes sure every single line of code was tested. TrueTime allows you to find out what the bottlenecks are in your code and helps you optimize and fine tune your code for maximum speed. These products may not come cheap, but they're worth every penny.

Zlib and Info-Zip

In previous releases of the Xceed Zip Compression Library before version 4.0, the zip file format handling routines, as well as the compression and decompression routines used to perform the major zip archive operations were based on the work by the Info-Zip group. We wish to thank the members of the Info-Zip group for the outstanding contribution that they have provided the software community.

As of the Xceed Zip Compression Library v4.0 release, all zip file format handling routines used by Xceed Zip are based on Xceed Software design and programming. Everything is written entirely in threadsafe, object-oriented Visual C++ 6.0 code.

The compression and decompression algorithms used by Xceed Zip are based on the Zlib library, written by Jean-Loup Gailly and Mark Adler. Zlib is a publicly available library that includes source code for the "deflate" compression and decompression algorithm. You can download it from the Zlib web site (if you are interested in it, we suggest you do a search on yahoo.com in order to make sure you have the latest web site address).

There are no extra charges or costs due to the use of the Zlib code in the Xceed Zip Compression Library.

Blue Sky

This help documentation was created with RoboHelp Office. It is a very complete package that includes lots of time-saving tools, is very robust, and fully integrates with Microsoft Word in order to provide a familiar .

Absolute Packager

The Xceed Zip Compression Library is packaged with a self-extracting zip file created by Absolute Packager, one of our own products. Absolute Packager itself internally uses the Xceed Zip Compression Library v3.5 and Xceed Zip Self-Extractor Module!

Xceed Software's address, phones and email

You can contact us through any of the following:

Internet info@xceedsoft.com (general inquiries)
support@xceedsoft.com (technical support)

Web site www.xceedsoft.com

Telephone (450) 442-2626

Orders (800) 865-2626

Fax (450) 442-4604

Mail Xceed Software Inc.
1555 Boul. Jean-Paul Vincent
Suite 180
Longueuil, Québec
Canada J4N 1L6

XCEED SOFTWARE INC.
1555 Boul. Jean-Paul Vincent
Suite 180
Longueuil, Québec
Canada J4N 1L6

Tel: (800) 865-2626 (Orders)
Tel: (450) 442-2626 (Tech)
Fax: (450) 442-4604

Fax Back Order Form

(450) 442-4604

Print Form

Please fill in the form below and send back for immediate service. **Please print clearly.**

Registration information			
Name/Contact:			
Company:			
Street:			
City:	State/Prov.:	Zip/Postal Code:	
Country:	Tel.:	Fax:	
Internet email:			
Where did you first hear about Xceed?			

Description	Price (US$)	Units	Total
Xceed Zip Compression Library v4.0	$299.95 US		$
Xceed Zip Self-Extractor Module (Requires Xceed Zip Compression Library)	$199.95 US		$
Combo Package (Includes both Xceed Zip Compression Library v4.0 and Xceed Zip Self-Extractor Module)	$399.95 US		$
Shipping and handling (Circle desired option)			
Canada	Mail $5 US, Mail+Email $5 US, FedEx $10 US		$
United States	Mail $10 US, Mail+Email $10 US, FedEx $20 US		$
Other	Mail $15 US, Mail+Email $15 US, FedEx $45 US		$
Taxes			
Sub-Total			$
Canadian residents add 7% GST, QC residents add 7.5% PST			$
		Total:	$

Payment (Sorry, no POs or Bank transfers accepted)

[ ] VISA [ ] MasterCard [ ] AmEx	Credit Card Number:_____
[ ] US$ or Canadian$ Check	Expiration Date:_____ Name On Card: _____
[ ] US$ International Bank Draft	Signature: _____

Please note that your order will be charged in Canadian dollars at the current US dollar exchange rate.

Document revision 5400

How to order Xceed Zip

To place an order

Contact Xceed Software directly using the (800) 865-2626 order line, fax/mail an Order Form, or use the secure online ordering system at www.xceedsoft.com. See the order form for pricing information.

If you are an established distributor or reseller and you wish to carry Xceed products, please contact us directly.

When ordering, please follow these simple guidelines

- Print or type clearly. Do not assume we will know what city, state/province, or country you are in. This is especially important for foreign orders, where it is more difficult for us to guess what a city or state might be. For telephone or fax numbers, please include your country code, and/or make sure that the country is spelled out completely. Do not use abbreviations such as UK, DN, FR, etc. It will avoid confusion and greatly accelerate processing your order.
- We can only accept credit cards, US or Canadian checks, and international bank drafts or money orders. Please do not send EuroChecks or request bank transfers. All checks must be drawn on a US or Canadian bank.
- If you are sending a check or a bank draft, and this file is more than 6 months old (past September 1999), please make sure that the price has not changed before you order.
- If you send a bank draft, please do not have the bank send it automatically. You must include an order form with the bank draft.
- If you order by credit card, please make sure you include all the details, including the card number, the expiration date, and the complete name on the card. Do not send a fax/letter asking us to call you for the card number. Call us directly if you do not want to include the card number on your fax.
- Although we will always do our utmost to process your order as quickly as possible, it can sometimes take up to 48 hours. Please consider this when you place a rush order.
- Finally, please note the time before you call and check the time difference between your time zone and ours. We are located in the Eastern standard time zone.

These few guidelines will help assure that everyone will get their order processed quickly and correctly.

Upgrades

We periodically provide bug fixes and upgrades for our products. Always check our Web site at www.xceedsoft.com for the latest information on the Xceed Zip Compression Library and the Xceed Zip Self-Extractor Module.

Upgrade policy

Minor changes and bug fixes are free; major revisions carry an upgrade fee. If we release a major revision within 60 days of your purchase, and it is identically priced, we will provide the upgrade free of charge. Shipping and handling are always extra. We do not automatically inform you of minor fixes, only major revisions.

Upgrade pricing and purchasing

Purchasing an upgrade entitles you to 1 year's worth of unlimited minor and major upgrades. A year's worth of upgrades for the Xceed Zip Compression Library costs $99 US.

The upgrade price applies to upgrades from any previous version of the Xceed Zip Compression Library or Xceed Zip Self-Extractor Module.

All upgrades and discounts must be obtained directly from Xceed Software Inc. See the How to Order topic.

Upgrade prices are subject to change without any prior notice.

Technical support

Before you contact us

In order to save everyone some time, we'd like you to do a few things before you contact us. We know the urge to call is great, but we'd appreciate it if you double check this list:

- Have you consulted the Windows Help manual?
- Have you used the search feature of the manual?
- Have you looked at the sample application sources?

If you got this far, it must be a tough one. Maybe it's time to tell us about your problem. Start by noting down the language you are using, the version of the Xceed Zip Compression Library, the exact error message (if any) or error code values you are getting, the operating system you are running your project on and a brief description of what you are trying to do. It would be a good idea to also note down exactly when the problem occurs.

How to contact us

The preferred method of contacting us for technical support is on the Internet. You can send any questions or problems to our Internet address: **support@xceedsoft.com**. Do not leave questions in newsgroups or on CompuServe forums as they probably won't be answered.

We also offer free telephone support on an "as available" basis, Monday through Friday usually from 9 a.m. to 5 p.m. Eastern standard time. Please note the time before you call and check the time difference between your time zone and ours. Sorry, we cannot return international calls.

Although there is currently no expiration date on free support, we reserve the right to restrict it to the first 90 days from the time of purchase, without any prior notice.

Year 2000 compliance

All of Xceed Software's active products have been tested and conform to Year 2000 compliance criteria. Xceed Software products have no inherent dependencies on date/time, and to the best of our knowledge, will not be adversely affected by the internal computer clock setting prior to, during and after January 1, 2000.

All future software upgrades of the products, listed below, shall be placed under the scrutiny of Year 2000 compliance. Software tested includes:

- Xceed Absolute Packager v1.0 and up
- Xceed Zip Compression Library v3.0 and up
- Xceed Zip Self-Extractor Module v1.0 and up

Xceed Software does not provide contractual warranties or guarantees specific to Year 2000 readiness. Xceed Software product guarantees are set forth in the License Agreement section included in each of our manuals. We recommend that you read these agreements in order to fully understand your rights. Any information provided by Xceed Software regarding Year 2000 issues does not constitute an extension of any Xceed Software guarantee.

Please note that the Xceed Zip Compression Library and Xceed Zip Self-Extractor Module are software development tools. We have no control over the ultimate quality or accuracy of products developed using our tools. Therefore, Xceed Software provides no guarantee whatsoever in connection with Year 2000 compliance of any software, application, or operating system developed by third parties using our products.

Zip method

VB example Delphi example

Description

The Zip method compresses and stores files into a new or existing zip file.

The zip file to be written to must be specified by setting the ZipFilename property. When you run the Zip method, if the specified zip file does not exist, a new one will be created.

The files to zip up must be specified by setting the FilesToProcess property.

Declaration

```
Method Zip() As xcdError
```

Parameters

None

Return values

A return value of type xcdError is returned. Only the return value of xerSuccess indicates that the method has been completed successfully. See the error codes topic for a list of possible return values.

Remarks

If one or more files to zip up already exist in the zip file being written to, the files already in the zip file will be replaced with new ones - unless one of the filtering properties causes the file to be excluded from processing.

Applicable properties

Abort, BasePath, CompressionLevel, EncryptionPassword, FilesToProcess and the filtering properties, SplitSize, PreservePaths, ProcessSubfolders, the SkipIf* properties, TempFolder, UseTempFile, ZipFilename, SpanMultipleDisks, ExtraHeaders, ZipOpenedFiles and the Sfx* properties.

Events triggered

ZipPreprocessingFile, FileStatus, GlobalStatus, QueryMemoryFile, InsertDisk, DiskNotEmpty, ZipComment, ZippingMemoryFile, Warning, SkippingFile, InvalidPassword, and ProcessCompleted.

Related topics

If you want to create spanned zip files, refer to the SpanMultipleDisks property.

Unzip method

VB example Delphi example

Description

The Unzip method uncompresses files that are stored in a zip file and recreates them into the folder specified by the UnzipToFolder property.

The zip file to unzip from must be specified by setting the ZipFilename property.

The files to unzip must be specified by setting the FilesToProcess property. If the FilesToProcess property is left empty, then all the files in the zip file will be unzipped. Use the filtering properties when you need to limit processing to only files with specific attributes, sizes, dates, etc.

Declaration

```
Method Unzip() As xcdError
```

Parameters

None

Return values

A return value of type xcdError is returned. Only the return value of xerSuccess indicates that the method has been completed succesfully. See the error codes topic for a list of possible return values.

Remarks

None

Applicable properties

Abort, EncryptionPassword, FilesToProcess and the filtering properties, PreservePaths, ProcessSubfolders, the SkipIf* properties, UnzipToFolder, ZipFilename and ExtraHeaders.

Events triggered

InsertDisk, UnzipPreprocessingFile, SkippingFile, FileStatus, GlobalStatus, UnzippingMemoryFile, Warning, InvalidPassword, ZipComment and ProcessCompleted.

PreviewFiles method

Description

The PreviewFiles method provides the complete list of files that will end up being processed if a call is made to the Zip method with the current property settings. The PreviewFiles method also calculates and reports statistics on the entire group of files listed.

The files listed by the PreviewFiles method are determined by the same properties that affect the Zip method – namely the FilesToProcess, FilesToExclude, the Skip* properties and the filtering properties.

The PreviewingFile event is triggered for each file being listed. This event provides detailed file information.

The ProcessCompleted event provides the statistics for the entire group of files listed.

Declaration

```
Method PreviewFiles(bCalcCompSize As Boolean) As xcdError
```

Parameters

bCalcCompSize

> The bCalcCompSize parameter determines whether the PreviewFiles method should calculate and report the total compressed size of all the listed files. If the bCalcCompSize parameter is set to True, the calculation is performed and the result is provided by the ProcessCompleted event. This allows you to make accurate estimates of the number of floppy or other removable media disks required in order to zip these files in a spanned zip file. The calculation involves completely reading each file's data and compressing it to memory.

Return values

A return value of type xcdError is returned. Only the return value of xerSuccess indicates that the method has been completed succesfully. See the error codes topic for a list of possible return values.

Remarks

The PreviewFiles method works by creating a copy of the contents of the FilesToProcess property, expanding all wildcards, removing all the files that aren't currently accessible on the local machine or network, removing files that match entries in the FilesToExclude property and removing any files excluded by the filtering properties . The resulting list of files ends up being the same as the list of files that will be processed by the Zip method when it is called.

Applicable properties

Abort, BasePath, FilesToProcess and the filtering properties, PreservePaths and ProcessSubfolders.

Events triggered

PreviewingFile, ProcessCompleted

Related topics

The ListZipContents method is the equivalent of the PreviewFiles method, but for files already inside a zip file.

ListZipContents method

VB example Delphi example

Description

The ListZipContents method lists and provides detailed information on the files contained in a zip file. It also calculates and reports statistics on the entire group of files listed.

The zip file to list must be specified by the ZipFilename property.

The ListingFile event is triggered for each file being listed. This event provides detailed file information.

The ProcessCompleted event provides the statistics for the entire group of files listed.

If the FilesToProcess property is empty, all files in the zip file will be listed, except those that are excluded by the filtering properties.

Declaration

```
Method ListZipContents() As xcdError
```

Parameters

None

Return values

A return value of type xcdError is returned. Only the return value of xerSuccess indicates that the method has been completed succesfully. See the error codes topic for a list of possible return values.

Remarks

The files listed by the ListZipContents method can be limited by the FilesToProcess, FilesToExclude and the filtering properties – but are not affected by the SkipIf* properties. Therefore, if you don't use the SkipIf* properties, the ListZipContents method can be used to obtain the complete list of files that will end up being processed by a call to the Unzip method.

Applicable properties

Abort, ZipFilename, FilesToProcess, ProcessSubfolders and the filtering properties.

Events triggered

ListingFile, InsertDisk, GlobalStatus, ZipComment, Warning and ProcessCompleted

Related topics

The PreviewFiles method is the equivalent of the ListZipContents method, but for files on disk – not already in a zip file.

RemoveFiles method

VB example Delphi example

Description

The RemoveFiles method removes files from a zip file.

The zip file to remove files from must be specified by setting the ZipFilename property.

The files to remove must be specified by setting the FilesToProcess property. The FilesToExclude property and the filtering properties are also consulted to allow you to limit processing to only specific types of files.

The RemoveFiles method will not operate on spanned zip files.

Declaration

```
Method RemoveFiles() As xcdError
```

Parameters

None

Return values

A return value of type xcdError is returned. Only the return value of xerSuccess indicates that the method has been completed succesfully. See the error codes topic for a list of possible return values.

Remarks

Be careful when the ProcessSubfolders property is set to true – specifying "*" in FilesToProcess will in this case erase all files from the zip file.

Applicable properties

Abort, FilesToProcess and the filtering properties, ProcessSubfolders, TempFolder and the UseTempFile property which must be set to True.

Events triggered

RemovingFile, GlobalStatus

Related topics

Related topics

TestZipFile method

Description

The TestZipFile method checks the integrity of a zip file, and can also check to make sure all files inside the zip file can be correctly unzipped back to their original uncompressed form without loss of information.

The Warning event is triggered whenever an error is detected in the zip file structure.

The TestZipFile method's return value provides you with the conclusion on the integrity of the zip file.

Declaration

```
Method TestZipFile(bCheckCompressedData As Boolean) As xcdError
```

Parameters

Parameter	Description
bCheckCompressedData	Determines whether or not each file's compressed data should be checked for errors. Setting this property to True will cause Xceed Zip to uncompress each file's data to memory and verify that the file's stored CRC corresponds with the uncompressed data's CRC. Setting this property to False will cause Xceed Zip to skip the compressed data test and report only zip file structure problems. This parameter is optional. If it is not supplied, it the default value is True.

Return values

A return value of type xcdError is returned. Only the return value of xerSuccess indicates that the method has been completed succesfully. See the error codes topic for a list of possible return values.

Remarks

If you are trying to determine the integrity of the individual files inside the zip file, as opposed to the integrity of the entire zip file, use the bFileCompleted parameter of the FileStatus event which is also generated when you run the TestZipFile method. When a FileStatus event is triggered with bFileCompleted = True, you know that the current file being tested is correctly unzippable.

You cannot set the FilesToProcess property to only test certain files in the zip file. The TestZipFile method tests all the files in a zip file when the bCheckCompressedData parameter is set to True.

Applicable properties

Abort, EncryptionPassword, ZipFilename

Events triggered

InvalidPassword, InsertDisk, SkippingFile, TestingFile, FileStatus, GlobalStatus, Warning and ProcessCompleted

AddFilesToProcess method

Description

The AddFilesToProcess method adds an item to the FilesToProcess property's list of files to process.

AddFilesToProcess automatically handles separating items with the CR+LF characters. In fact, it performs the same work as the following line of code:

```
XceedZip1.FilesToProcess = XceedZip1.FilesToProcess + sFileMask + Chr(13) +
    Chr(10)
```

Declaration

```
Method AddFilesToProcess(sFileMask As String)
```

Parameters

Parameter	Description
sFileMask	The item to add to the FilesToProcess property's list of files.

Return values

None

Applicable properties

FilesToProcess

Events triggered

None

AddFilesToExclude method

Description

The AddFilesToExclude method adds an item to the FilesToExclude's list of files to process.

AddFilesToExclude automatically handles separating items with the CR+LF characters. In fact, it performs the same work as the following line of code:

```
XceedZip1.FilesToExclude = XceedZip1.FilesToExclude + sFileMask + Chr(13) +
    Chr(10)
```

Declaration

```
Method AddFilesToExclude(sFileMask As String)
```

Parameters

Parameter	Description
sFileMask	The item to add to the FilesToExclude property's list of files.

Return values

None

Applicable properties

FilesToExclude

Events triggered

None

Abort property

Description

The Abort property allows you to stop the execution of a currently running Xceed Zip method.

At any time during the execution of an Xceed Zip method, setting the Abort property to True will cause the currently executing method to abort and return the xerUserAbort value, which indicates that the abort was successful. The Abort property is automatically set back to False whenever a new method is called.

In order to properly terminate the current operation, the response to the Abort property is not always instantaneous.

This property is not browsable by an object browser.

Data type

Boolean

Default value

False

Applicable methods

All methods except AddFilesToProcess and AddFilesToExclude

Related topics

To find out what the current operation being executed is, use the CurrentOperation property.

BasePath property

Example

Description

The BasePath property allows you to specify the starting path for all entries in the FilesToProcess and FilesToExclude properties that have been specified using relative paths.

If the BasePath property is left empty, relative paths specified in the FilesToProcess and FilesToExclude properties will be relative to the application's current directory, which is not recommended.

This property is particularly useful for controlling how paths are stored in the zip file.

Data type

String

Default value

Empty

Applicable methods

Zip

Related topics

If you want more control over how path and/or filenames are stored in the zip file, you can use the ZipPreprocessingFile event to change stored path and filenames to whatever you want.

If you don't want any path information to be stored in the zip file, you can use the PreservePaths property.

CompressionLevel property

Description

The CompressionLevel property controls the amount of compression to be applied to files being zipped. The greater the amount of compression applied, the greater the time it takes to perform the compression. The CompressionLevel property affects the Zip method only.

Data type

xcdCompressionLevel

Possible values

Value	Meaning
xclNone (0)	No compression. Files are stored in the zip file as raw data.
xclLow (1)	Minimum compression. This setting takes the least amount of time to compress files. When compared to the XcdMedium setting, this setting usually gives noticeably faster compression times in exchange for about 1 to 15% larger compressed files.
xclMedium (6)	Normal compression. This is the best balance between the time it takes to compress a file and the compression ratio achieved.
xclHigh (9)	Maximum compression. This setting achieves the best compression ratios that the Zip deflate compression algorithm is capable of producing. When compared to the xcdMedium setting, this setting significantly increases compression time in exchange for about 1-3% smaller compressed files. We recommend that you use this setting only when you really need to achieve the smallest possible files and when compression time is unimportant.

Default value

xclMedium

Applicable methods

Zip

EncryptionPassword property

Description

The EncryptionPassword property contains the case-sensitive password to be used for encrypting or decrypting files.

The password specified in the EncryptionPassword property applies to all the files being zipped, unzipped or tested. To be able to specify different passwords for each file, use the ZipPreprocessingFile or UnzipPreprocessingFile events which are triggered for each file before it is zipped, unzipped or tested.

If the EncryptionPassword property is left empty before executing the Unzip or TestZipFile methods, and no password is provided using the UnzipPreprocessingFile event, then no encrypted files will be unzipped or tested. The SkippingFile event is triggered for each file skipped because of a missing or incorrect password.

If the EncryptionPassword property is empty before executing the Zip method, and no password is provided using the ZipPreprocessingFile event, then any newly zipped files will not be encrypted. Files that are already in the zip file will remain in their current state (either non-encrypted or encrypted) and must be rezipped in order to change them from non-encrypted to encrypted files and vice-versa.

Passwords can be up to 80 characters long.

Data type

String

Default value

Empty

Remarks

The encryption algorithm used by the Xceed Zip Compression Library is the same one used by the PKZip 2.04g program. This type of encryption can resist attacks by amateurs if the password is well chosen and long enough (at least 16 characters) but it will probably not resist attacks by determined users or experts. Therefore, we suggest that you always use long passwords, and when possible, do not rely solely on this encryption system to protect sensitive data.

Applicable methods

Zip, Unzip, TestZipFile

RequiredFileAttributes property

Example

Description

The RequiredFileAttributes property lets you limit the processing of files to only those files that have specific file attributes on. This property affects the list of files specified in the FilesToProcess property.

Files that do not have the required file attributes will be tagged as "excluded" when the ZipPreprocessingFile or UnzipPreprocessingFile event is triggered, and will not be processed unless you change the value of the *excluded* parameter to False during the event.

To specify multiple file attributes that must be on, set the RequiredFileAttributes property equal to the sum of the values for each required file attribute.

Data type

xcdFileAttributes

Possible values

Value	Meaning
xfaNone (0)	No specific attributes must be on in order to process files.
xfaReadOnly (1)	Read-only attribute required.
xfaHidden (2)	Hidden attribute required.
xfaSystem (4)	System attribute required.
xfaVolume (8)	Volume Label attribute required.
xfaFolder (16)	Folder / directory attribute required.
xfaArchive (32)	Archive bit attribute required.

Default value

xfaNone

Applicable methods

Zip, Unzip, ListZipContents, PreviewFiles, RemoveFiles

Related topics

The ExcludedFileAttributes property lets you limit the processing of files to only those files that don't have specific file attributes on, as opposed to only processing files that have these attributes on.

If you want to create your own, special filters in order to limit which files are processed, you can use the ZipPreprocessingFile or UnzipPreprocessingFile events to specifically exclude files (just set the *excluded* parameter whenever you want to exclude a file).

ExcludedFileAttributes property

Example

Description

The ExcludedFileAttributes property lets you limit the processing of files to only those files that do not have specific file attributes on. This property affects the list of files specified in the FilesToProcess property.

Files that have the file attributes specified with the ExcludedFileAttributes property will be tagged as "excluded" when the ZipPreprocessingFile or UnzipPreprocessingFile event is triggered, and will not be processed unless you change the value of the *excluded* parameter to False.

To specify multiple file attributes that you do not want to process, set the ExcludedFileAttributes property equal to the sum of the values for each file attribute.

Data type

xcdFileAttributes

Possible values

Value	Meaning
xfaNone (0)	No specific attributes must be off in order to process files.
xfaReadOnly (1)	Read-only attribute must be off.
xfaHidden (2)	Hidden attribute must be off.
xfaSystem (4)	System attribute must be off.
xfaVolume (8)	Volume Label attribute must be off.
xfaFolder (16)	The folder / directory attribute must be off.
xfaArchive (32)	Archive bit attribute must be off.
xfaAlias (64)	Alias attribute must be off.
xfaCompressed (2048)	Compressed attribute must be off.

Default value

xfaVolume + xfaFolder

Applicable methods

Zip, Unzip, ListZipContents, PreviewFiles, RemoveFiles

Related topics

The RequiredFileAttributes property lets you limit the processing of files to only those files that have specific file attributes on, as opposed to only processing files that don't have these attributes on.

If you want to create your own, special filters in order to limit which files are processed, you can use the ZipPreprocessingFile or UnzipPreprocessingFile events to specifically exclude files (just set the *excluded* parameter whenever you want to exclude a file).

FilesToProcess property

Description

The FilesToProcess property is used to provide a list of files to be processed by the Zip, Unzip, RemoveFiles, PreviewFiles, ListZipContents and TestZipFile methods.

Wildcards can be used to match multiple files. If you set the ProcessSubfolders property to True, all subfolders will also be scanned for matching files and folders.

When using the **Zip** or **PreviewFiles** methods, the FilesToProcess property must contain a list of files or folders to zip up (or preview) that match files accessible on the current machine or network. For these methods, the files to process can be specified either with absolute paths, relative paths or without paths. The manner that files are specified in the FilesToProcess property determines what portion of the path will be stored in the zip file along with the filename. See the controling how paths are stored topic for details.

When using the **Unzip**, **RemoveFiles**, **ListZipContents** or **TestZipFile** methods, the filenames you provide to the FilesToProcess property must match with the path and filename information stored in the Zip file. If you are unsure of what path information is stored in the zip file, set the PreservePaths property to False. This will cause all path information stored in the zip file to be ignored, and will allow you to specify only filenames in the FilesToProcess property.

If you leave the FilesToProcess property empty and use the Unzip, ListZipContents or the TestZipFile methods, all the files in the zip file will be processed.

Data type

String. The FilesToProcess property is a single string that can contain multiple files and wildcards to process. When specifying multiple files to process, each file must be separated with the CR character (ASCII 13), the LF character (ASCII 10), the CR+LF characters or the vertical bar "|" character. For cleaner code, use the AddFilesToProcess method when you need to specify multiple files in the FilesToProcess property – it automatically handles the task of separating items with the return character.

Default value

Empty

Remarks

When using relative paths to zip or preview files, the paths must be relative to the application's current directory, or the folder specified in the BasePath property if it has been set.

Applicable methods

Zip, Unzip, ListZipContents, PreviewFiles, RemoveFiles

Related topics

You can use the FilesToExclude property if you want to exclude specific filenames or paths from the list of files you have specified in the FilesToProcess property.

To exclude files based on criteria other than filenames, refer to the filtering properties.

FilesToExclude property

Example

Description

The FilesToExclude property lets you exclude specific files or wildcards from the list of files to process provided to the FilesToProcess property.

Files to process that match the files provided to the FilesToExclude property will be tagged as "excluded" when the ZipPreprocessingFile or UnzipPreprocessingFile event is triggered, and will not be processed unless you change the value of the *excluded* parameter to False.

The FilesToExclude property is used the same way as the FilesToProcess property:

- When using the Zip or the PreviewFiles method, files can be specified with absolute paths, relative paths or without paths.
- When using the Unzip, RemoveFiles, ListZipContents or the TestZipFile method, the filenames you provide to the FilesToExclude property must match with the path and filename information stored in the zip file.
- When specifying multiple files to exclude, each file must be separated with the carriage return (ASCII 13), line feed (ASCII 10) character or the "|" character.
- There's an AddFilesToExclude method equivalent of the AddFilesToProcess method.
- You can use the "*" and "?" wildcards to exclude multiple files.

Data type

String

Default value

Empty

Remarks

When using relative paths, they must be relative to the application's current directory, or the folder specified in the BasePath property if it has been set.

Applicable methods

Zip, Unzip, ListZipContents, PreviewFiles, RemoveFiles

Related topics

To exclude files based on criteria other than filenames, refer to the filtering properties.

MinDateToProcess property

Example

Description

The MinDateToProcess property is a filtering property that lets you limit the processing of files to only those files that have a date and time stamp larger than or equal to a specified date and time. This property affects the list of files specified in the FilesToProcess property.

Files that have a date and time smaller than the date and time specified in the MinDateToProcess property will be tagged as "excluded" when the ZipPreprocessingFile or UnzipPreprocessingFile event occurs, and will not be processed unless you change the value of the *excluded* parameter to False.

This property can be used in conjunction with other filtering properties. In particular, it can be used in conjunction with the MaxDateToProcess property in order to provide a range of date and time stamps that should be processed.

Data type

Date

Possible values

Dates can range from 01/01/1900 to 31/12/9999

Default value

01/01/1980 (All files qualify for processing)

Applicable methods

Zip, Unzip, ListZipContents, PreviewFiles, RemoveFiles

Related topics

The SkipIfOlderDate property lets you skip files about to be zipped that are older than the files already in the zip file, or skip files about to be unzipped that are older than the files already existing in the unzipping location.

MaxDateToProcess property

Example

Description

The MaxDateToProcess property is a filtering property that lets you limit the processing of files to only those files that have a date and time stamp smaller than or equal to a specified date and time. This property affects the list of files specified in the FilesToProcess property.

Files that have a date and time larger than the date and time specified in the MaxDateToProcess property will be tagged as "excluded" when the ZipPreprocessingFile or UnzipPreprocessingFile event occurs, and will not be processed unless you change the value of the *excluded* parameter to False.

This property can be used in conjunction with other filtering properties. In particular, it can be used in conjunction with the MinDateToProcess property in order to provide a range of date and time stamps that should be processed.

Data type

Date

Possible values

Dates can range from 01/01/1900 to 31/12/9999

Default value

31/12/9999 (All files qualify for processing)

Applicable methods

Zip, Unzip, ListZipContents, PreviewFiles, RemoveFiles

Related topics

The SkipIfOlderDate property lets you skip files about to be zipped that are older than the files already in the zip file, or skip files about to be unzipped that are older than the files already existing in the unzipping location.

MinSizeToProcess property

Example

Description

The MinSizeToProcess property is a filtering property that lets you limit the processing of files to only those files that are larger than or equal to a specified file size. This property affects the list of files specified in the FilesToProcess property.

Files that have a smaller file size (in bytes) than the amount specified MinSizeToProcess property will be tagged as "excluded" when the ZipPreprocessingFile or UnzipPreprocessingFile event occurs, and will not be processed unless you change the value of the *excluded* parameter to False.

This property can be used in conjunction with other filtering properties. In particular, it can be used in conjunction with the MaxSizeToProcess property in order to provide a range of file sizes that should be processed.

Data type

Long Integer

Possible values

0 to 4294967295 (or 0 to 2147483647 for languages with signed long integers such as VB)

Default value

0 (No lower file size limit, all files qualify for processing)

Applicable methods

Zip, Unzip, ListZipContents, PreviewFiles, RemoveFiles

Related topics

None

MaxSizeToProcess property

Example

Description

The MaxSizeToProcess property is a filtering property that lets you limit the processing of files to only those files that are smaller than or equal to a specified file size. This property affects the list of files specified in the FilesToProcess property.

Files that have a larger file size (in bytes) than the amount specified MaxSizeToProcess property will be tagged as "excluded" when the ZipPreprocessingFile or UnzipPreprocessingFile event occurs, and will not be processed unless you change the value of the *excluded* parameter to False.

Setting this property to 0 will eliminate checking for a maximum size.

This property can be used in conjunction with other filtering properties. In particular, it can be used in conjunction with the MinSizeToProcess property in order to provide a range of file sizes that should be processed.

Data type

Long Integer

Possible values

0 to 4294967295 (or 0 to 2147483647 for languages with signed long integers such as VB)

Default value

4294967295 (No upper file size limit, all files qualify for processing)

Applicable methods

Zip, Unzip, ListZipContents, PreviewFiles, RemoveFiles

SplitSize property

Description

The SplitSize property lets you create zip files in multiple parts located in the same place that are no larger than a specific file size. The resulting parts are identical to spanned zip files, except that they are not located on different removable media disks and each part has a different filename.

For multi-part zip files, the first part of the zip file usually has the ".zip" filename extension, while the remaining parts have extensions named ".z01", ".z02", and so on.

To create zip files in multiple parts, set the SplitSize property to the maximum size, in bytes, that you want each part to have, and run the Zip method. Zipping will start, as usual, in the target zip file specified in the ZipFilename property. If the resulting zip file reaches the maximum zip file size, it will be closed and zipping will continue in the next zip file part, with the filename extension ".z01". None of the parts will exceed the number of bytes specified in the SplitSize property, but some parts may be smaller than the maximum size.

Setting the SplitSize property to 0 disables the creation of multi-part zip files in the same location, but does not affect the outcome of the SpanMultipleDisks property.

Data type

Long Integer

Possible values

0 to 4294967295 (or 0 to 2147483647 for languages with signed long integers such as VB)

Default value

0

Remarks

You can use the SplitSize property in conjunction with the SpanMultipleDisks property to create spanned zip files that are limited to a specific size and do not necessarily fill up the entire removable media disk.

Applicable methods

Zip

Related topics

None

PreservePaths property

Description

The PreservePaths property allows you to determine whether path information is stored in (or retrieved) from the zip file.

When using the Zip method, setting PreservePaths to False will cause all path information associated with a file to be discarded, and will cause only the file's filename to be stored in the zip file. Setting PreservePaths to True will cause path information to be stored in the zip file along with each file's filename. See the controling how paths are stored topic for more information on storing paths when using the Zip method.

When using the Unzip method, or other methods that require files to be specified in the FilesToProcess property, setting PreservePaths to False will cause Xceed Zip to ignore any path information stored in the zip file. When this is done, you must specify the files to process only with filenames (no path information must be specified) otherwise no files will be matched for processing. When PreservePaths is set to True, the FilesToProcess property must contain path and filename information that matches path and filename information stored in the zip file.

Also when using the unzip method: If PreservePaths is set to True, pathnames that have been stored in the zip file along with each file's filename will be used to determine where the files will be unzipped to. The stored pathnames and files will be created inside the destination unzipping folder specified with the UnzipToFolder property. If PreservePaths is set to False, all files will be directly unzipped (without any subfolders being created) to the destination unzipping folder.

Data type

Boolean

Default value

True

Applicable methods

Zip, Unzip, PreviewFiles

Related topics

None

ProcessSubfolders property

Description

The ProcessSubfolders property is used to determine whether subfolders are processed or not.

When the ProcessSubfolders property is set to True, you can zip or unzip entire folder structures. For example: Specifying "C:\WINDOWS*" in the FilesToProcess property and setting the ProcessSubfolders property to True will cause the Zip method to zip up the entire contents of the "C:\WINDOWS" folder, including the SYSTEM folder and all other files and folders that can be found in there.

When the ProcessSubfolders property is set to False, subfolders will not be processed.

Data type

Boolean

Default value

True

Remarks

Use "*" to match all files in a given folder. Old-style DOS wildcards such as "*.*" do not work in 32-bit operating systems (Windows 95/98/NT) the way they do in DOS. In Windows 95/98/NT, "*.*" will only match files that have a "." in them.

Applicable methods

Zip, Unzip, ListZipContents, PreviewFiles, RemoveFiles

Related topics

The controling how paths are stored topic provides information on how to control what portions of a file's path (if any) are stored in the zip file.

SkipIfExisting property

Description

The SkipIfExisting property allows you to process only files that do not already exist in the destination unzipping folder or, when zipping, the destination zip file.

When using the Zip method, setting SkipIfExisting to True will cause files that already exist in the destination zip file to be skipped (excluded from processing), thus leaving the already existing file(s) intact.

The same idea applies when using the Unzip method: setting SkipIfExisting to True will cause files that already exist in the destination unzipping folder to be skipped.

Files that have been excluded because of the SkipIfExisting property will be tagged as "excluded" when the ZipPreprocessingFile or UnzipPreprocessingFile event is triggered, and will not be processed unless you change the value of the *excluded* parameter to False.

Data type

Boolean

Default value

False

Applicable methods

Zip, Unzip

Related topics

See the SkipIfNotExisting, SkipIfOlderDate and SkipIfOlderVersion properties for variations on this theme.

SkipIfNotExisting property

Description

The SkipIfNotExisting property allows you to process only files that already exist in the destination unzipping folder or, when zipping, the destination zip file.

When using the Zip method, setting SkipIfNotExisting to True will cause files that do not already exist in the destination zip file to be skipped (excluded from processing). This results in a zipping process that only replaces files that exist in the zip file but does not add any new files.

The same idea applies when using the Unzip method: setting SkipIfExisting to True will cause files that do not already exist in the destination unzipping folder to be skipped.

Files that have been excluded because of the SkipIfNotExisting property will be tagged as "excluded" when the ZipPreprocessingFile or UnzipPreprocessingFile event is triggered, and will not be processed unless you change the value of the *excluded* parameter to False.

Data type

Boolean

Default value

False

Applicable methods

Zip, Unzip

Related topics

See the SkipIfExisting, SkipIfOlderDate and SkipIfOlderVersion properties for variations on this theme.

SkipIfOlderDate property

Description

The SkipIfOlderDate property allows you to process only files that have a date and time stamp greater or equal to the date and time stamp of corresponding files already in the destination unzipping folder or, when zipping, the destination zip file.

When using the Zip method, setting SkipIfOlderDate to True will compare the date and time stamps of files about to be zipped with the date and time stamp of corresponding files already in the destination zip file. Files with an older date and time stamp than those already in the zip file will be skipped (excluded from processing).

The same idea applies when using the Unzip method: setting SkipIfOlderDate to True will compare the date and time stamps of files about to be unzipped with the date and time stamp of corresponding files already in the destination unzipping folder. Files with an older date and time stamp than those already in the destination unzipping folder will be skipped.

Files that have been excluded because of the SkipIfOlderDate property will be tagged as "excluded" when the ZipPreprocessingFile or UnzipPreprocessingFile event is triggered, and will not be processed unless you change the value of the *excluded* parameter to False.

Data type

Boolean

Default value

False

Applicable methods

Zip, Unzip

Related topics

See the SkipIfExisting, SkipIfNotExisting and SkipIfOlderVersion properties for variations on this theme.

SkipIfOlderVersion property

Description

The SkipIfOlderVersion property allows you to process only files that have a version number greater or equal to the version number of corresponding files already in the destination unzipping folder or, when zipping, the destination zip file. The version number is determined by consulting the version information resource usually embedded in executable files of various kinds, such as .EXEs, .DLLs and .OCXs.

When using the Zip method, setting SkipIfOlderVersion to True will compare the version numbers of files about to be zipped with the version numbers corresponding files already in the destination zip file. Files with an older version number (or no version number) than those already in the zip file will be skipped (excluded from processing).

The same idea applies when using the Unzip method: setting SkipIfOlderVersion to True will compare the version numbers of files about to be unzipped with the version numbers of corresponding files already in the destination unzipping folder. Files with an older version number (or no version number) than those already in the destination unzipping folder will be skipped.

Files that have been excluded because of the SkipIfOlderVersion property will be tagged as "excluded" when the ZipPreprocessingFile or UnzipPreprocessingFile event is triggered, and will not be processed unless you change the value of the *excluded* parameter to False.

Data type

Boolean

Default value

False

Applicable methods

Zip, Unzip

Related topics

See the SkipIfExisting, SkipIfNotExisting and SkipIfOlderDate properties for variations on this theme.

SpanMultipleDisks property

Description

The SpanMultipleDisks property allows you to control whether spanned zip files should be created if the drive or disk where the zip file is being written gets full. This property applies only when using the Zip method.

Data type

xcdDiskSpanning

Possible values

Value	Meaning
xdsNever (0)	If the drive or disk where the zip file is being written gets full, zipping will be aborted and the xerWriteZipFile error will be returned.
xdsAlways (1)	If the drive or disk where the zip file is being written gets full, the InsertDisk event will be triggered, allowing your application to perform various actions, including the possibility to query for a new disk to be inserted.
xdsAutoDetect (2)	If the drive or disk where the zip file is being written is considered to be a removable media device by Windows, the InsertDisk event will be triggered when the drive gets full. If it is not considered to be a removable media device by Windows, then zipping will be aborted and the xerWriteZipFile error will be returned. Note: CD-ROMs, floppy disks, Superdisks™, Zip™ and Jazz™ drives are all considered removable media devices, but not when they are accessed through a network. If you are using one of these devices through a network, use the xdsAlways setting instead of the xdsRemovableDrivesOnly setting.

Default value

xdsAutoDetect

Remarks

When SpanMultipleDisks is set to xdsAlways or xdsRemovableDrivesOnly and that a removable drive is being written to, you can only use the Zip method to create a new spanned zip file. Once the spanned zip file has been created and contains files, no more files can be added to, or replaced in it.

When creating spanned zip files, the zip format dictates that the directory of files contained in the zip file is located on the last disk of the disk set. That is why the last disk must always be inserted before the zip file can be unzipped, listed or verified with the Unzip, ListZipContents or TestZipFile methods. If the last disk is not present in the drive when trying to process a spanned zip file, the InsertDisk event will be triggered so your application can query for the last disk to be inserted. In some cases, such as when there are very many files in the spanned zip file, the directory may be both on the last disk, and the next-to-last disk. Xceed Zip will automatically trigger the InsertDisk twice, once for each disk.

When creating spanned files on devices considered to be removable media by Windows, the volume label of the media is changed to indicate the number of the diskette in the zip file set.

Setting the UseTempFile property to True will allow spanned zip files to be created more efficiently when writing to slow removable media – its use is recommended.

It is a good idea to inform users of your application to label diskettes when creating spanned zip files. A good time to do this is when the InsertDisk event is triggered and your application displays a dialog box requesting the next disk.

Applicable methods

Zip

Related topics

The SplitSize property lets you create spanned zip files that are located in the same place, as opposed to creating spanned zip files located on different disks. It also lets you limit the amount of data written to each disk, if you don't necessarily want to completely fill up each disk.

TempFolder property

Description

The TempFolder property lets you control the location where Xceed Zip will create a temporary file to work with when running the Zip or RemoveFiles methods.

The TempFolder property requires a path only (no filename) and does not require a trailing backslash.

If the TempFolder property is left empty, a temporary file will be created in the Windows temp folder.

The UseTempFile property allows you to control, in most situations, whether or not a a temporary file should be used.

Data type

String

Default value

Empty (use Windows temp folder)

Remarks

It is recommended that you specify a path for the TempFolder property by using an absolute path. If you use a relative path, be cautious because the path must be relative to the application's current directory – which often is tricky to work with.

Applicable methods

Zip, RemoveFiles

UseTempFile property

Description

The UseTempFile property influences whether or not the Zip method will use a temporary file when creating a new zip file or when writing to an existing zip file.

A temporary file is useful in order to help keep the zip file currently being written to in a consistent state in the event that the current operation fails, is abnormally aborted, or a system crash occurs. A temporary file is also useful when writing to zip files located on write-once media (CD-WORM) or across a network because the zip file is created in the temporary file (usually on a local drive) and written to the destination drive or network location only when the zipping operation is complete. A temporary file also speeds up the zipping operation when creating spanned zip files.

Setting the UseTempFile property to True will cause Xceed Zip to always create a temporary file. Setting it to False will cause Xceed Zip to avoid creating a temporary file if possible. Even if UseTempFile is set to False, a temporary file is always created when replacing a file in the destination zip file (for example, you zip up a file that already exists in the destination zip file, causing the file already in the zip file to be overwritten.)

Data type

Boolean

Default value

True

Remarks

A temporary file is always created when the RemoveFiles method is used. The UseTempFile property must therefore be set to True.

When an operation that causes a temporary file to be created has completed, the temporary file will be removed – even if an error occurs. However, if a system crash or reboot occurs before the operation completes, the temporary file will be left lying about, and must be removed manually.

Applicable methods

Zip, RemoveFiles

UnzipToFolder property

Description

The UnzipToFolder property is used by the Unzip method to determine where to unzip files to.

You can specify an absolute path, a relative path, or you can leave the UnzipToFolder property empty. If left empty, files will be unzipped into the application's current directory. If a relative path is used, the path must be relative to the application's current directory.

As described in the brief introduction to the zip format topic, files that are stored in a zip file can contain path information. This path information is always appended to the path indicated in the UnzipToFolder property unless the PreservePaths property has been set to False. In other words, if a file is stored in the zip file with a path and filename of "PICTURES\PICTURE1.BMP", and the UnzipToFolder property is set to "C:\INCOMING", then Xceed Zip will create a folder called "PICTURES" and unzip the file as "C:\INCOMING\PICTURES\PICTURE1.BMP". If PreservePaths is set to False, then Xceed Zip will unzip the file as "C:\INCOMING\PICTURE1.BMP".

The UnzipToFolder property requires a path only (no filename) and does not require a trailing backslash.

Data type

String

Default value

Empty

Remarks

The Unzip method will automatically create any folders that don't already exist in order to be able to unzip into the folder specified in the UnzipToFolder property.

Applicable methods

Unzip

ZipFilename property

Description

The ZipFilename property must specify the zip file to be used by the Xceed Zip component's methods.

You can specify the zip file name using an absolute path, a relative path, or no path at all. If no path is provided, the zip file will be created or searched for in the application's current directory. If a relative path and filename are used, the path must be relative to the application's current directory.

The zip file name can be any valid filename with any extension. The ".ZIP" filename extension is recommended because it makes the file easily identifiable as a zip file.

Examples of valid ZipFilename property settings are: "C:\OUTGOING\PICTURES.ZIP", "..\MY ZIP FILE.ZIP", "BACKUP.DAT", "\\MAINSERVER\PUBLICSHARE\DATA FILES\BACKUP 1491.ZIP" and "TEST".

Data type

String

Default value

Empty

Remarks

If this property is left empty, Xceed Zip methods will return the xerOpenZipFile error.

Applicable methods

All methods except AddFilesToProcess and AddFilesToExclude

CurrentOperation property

Description

The CurrentOperation property indicates what the current instance of the Xceed Zip control is doing.

For example, when you are zipping files, consulting the CurrentOperation property will yield the *xcoZipping* constant.

This property is read-only and is not browsable by an object browser.

Data type

xcdCurrentOperation

Possible values

Value	Meaning
xcoIdle (0)	No method currently executing
xcoPreviewing (1)	Executing the PreviewFiles method
xcoListing (2)	Executing the ListZipContents method
xcoZipping (3)	Executing the Zip method
xcoUnzipping (4)	Executing the Unzip method
xcoRemoving (5)	Executing the RemoveFiles method
xcoTestingZipFile (6)	Executing the TestZipFile method

Default value

xcdIdle

Applicable methods

All methods except AddFilesToProcess and AddFilesToExclude

ExtraHeaders property

Description

The ExtraHeaders property lets you control what type of supplemental file information you want stored in a zip file or retrieved from a zip file. This property applies to the Zip and Unzip methods only.

Data type

xcdExtraHeader

Possible values

Value	Meaning
xehPkwareFileTime (1)	Store/Retrieve extended time-stamp information about a file using a header format defined by PKWare®. Time-stamp information is stored in Windows NT format and includes the last modification date and time, the last accessed date and time and the creation date and time.
xehInfoZipFileTime (2)	Store/Retrieve extended time-stamp information about a file using a header format defined by the Info-Zip group. Time-stamp information is stored in Unix format and includes the last modification date and time, the last accessed date and time and the creation date and time.
xehSecurityDescriptor (4)	Store/Retrieve the Windows NT file security information header.
xehUnicode (8)	Store/Retrieve the Unicode filename header.

Default value

xehInfoZipFileTime + xehUnicode

Applicable methods

Zip, Unzip, Convert

ListingFile event

VB example Delphi example

Description

The ListingFile event is triggered once for each file in a zip file that is being listed as a result of calling the ListZipContents method.

The file information provided by the ListingFile event's parameters is not modifiable.

Declaration

```
Event ListingFile(sFilename As String, sComment As String, lSize As Long,
lCompressedSize As Long, nCompressionRatio As Integer, xAttributes As
xcdFileAttributes, lCRC As Long, dtLastModified As Date, dtLastAccessed As Date,
dtCreated As Date, xMethod As xcdCompressionMethod, bEncrypted As Boolean,
nDiskNumber As Long, bExcluded As Boolean, xReason As xcdSkippingReason)
```

Parameters

Parameter	Description
sFilename	The path and filename of the file being listed, exactly as it is stored in the zip file.
sComment	The file's comment.
lSize	The file's original uncompressed size in bytes.
lCompressedSize	The file's compressed size.
nCompressionRatio	The compression ratio achieved for this file.
xAttributes	The file's attributes.
lCRC	The file's uncompressed data CRC checksum.
dtLastModified	The last modification date and time stamp.
dtLastAccessed	The last accessed date and time stamp.
dtCreated	The creation date and time stamp.
xMethod	The compression method
bEncrypted	Is this an encrypted file?
nDiskNumber	When dealing with spanned zip files, the disk this file is located on.
bExcluded	Would this file be processed or excluded if the Unzip method were to be called with the same property settings as when the ListZipContents method was called?
xReason	If it will be excluded, why?

Applicable methods

ListZipContents

PreviewingFile event

Description

The PreviewingFile event is triggered once for each file that is being listed as a result of calling the PreviewFiles method.

The file information provided by the PreviewingFile event's parameters is not modifiable.

Declaration

```
Event PreviewFiles(sFilename As String, sSourceFilename As String, lSize As Long,
xAttributes As xcdFileAttributes, dtLastModified As Date, dtLastAccessed As Date,
dtCreated As Date, bExcluded As Boolean, xReason As xcdSkippingReason)
```

Parameters

Parameter	Description
sFilename	The path and filename of the file being listed, exactly as it would be stored in a zip file if the Zip method were to be executed instead of the PreviewFiles method.
sSourceFilename	The real path and filename of the file being listed.
lSize	The file's size in bytes.
xAttributes	The file's attributes.
dtLastModified	The last modification date and time stamp.
dtLastAccessed	The last accessed date and time stamp.
dtCreated	The creation date and time stamp.
bExcluded	Would this file be processed or excluded if the Zip method were to be called with the same property settings as when the PreviewFiles method was called?
xReason	If it will be excluded, why?

Applicable methods

PreviewFiles

InsertDisk event

Description

The InsertDisk event is triggered whenever a new or different disk is required to be inserted in a drive so that the current operation can proceed.

When using the Zip method and the SpanMultipleDisks property is set to span disks, this event is triggered when the current disk being written to gets full. This event allows you to prompt the end-user to insert a new disk. The nDiskNumber parameter provides the number of the current disk being requested. After the user has been queried to insert a new disk, the xAction parameter, which is modifiable, must be set to xacRetry. If it is left to the default value, which is xacAbort, the zipping operation will be aborted. Your application should inform the user to label each disk with the appropriate disk number as soon as it has been filled. If the same disk is still in the drive after the xAction parameter has been set to xacRetry and the InsertDisk event has completed, the InsertDisk event will be triggered again.

When using the Unzip, ListZipContents or TestZipFile methods and a spanned zip file is being read, this event will be triggered whenever data must be read from a disk in the disk set other than the one currently in the drive. The nDiskNumber parameter provides the number of the current disk being requested. After the user has been queried to insert a new disk, the xAction parameter, which is modifiable, must be set to xacRetry. If it is left to the default value, which is xacAbort, the unzipping operation will be aborted. If the wrong disk is inserted, or the same disk is still in the drive after the xAction parameter has been set to xacRetry and the InsertDisk event has completed, the InsertDisk event will be triggered again.

Declaration

```
Event InsertDisk(nDiskNumber As Long, bDiskInserted As Boolean)
```

Parameters

Parameter	Description
nDiskNumber	The number of the requested disk. If nDiskNumber = xcdLastDisk (0) then the last disk of the disk set is requested.
bDiskInserted	Set this to True to confirm that you have requested the insertion of the appropriate disk. If you leave it to its default value of False, the current operation will terminate and return xerInsertDisk abort.

Applicable methods

Zip, Unzip, ListZipContents, TestZipFile

ZipPreprocessingFile event

Description

The ZipPreprocessingFile event provides the opportunity to view and change a file's information before it is zipped, and to make the final decision about whether the file should be zipped or excluded from being zipped. This event is triggered right before a file is opened for zipping.

Not all file information provided by the ZipPreprocessingFile event's parameters is modifiable. The table below provides a list of parameters provided by ZipPreprocessingFile and shows you which ones can be modified.

Declaration

Parameters in bold are modifiable

```
Event ZipPreprocessingFile(sFilename As String, sComment As String, sSourceFilename
As String, lSize As Long, xAttributes As xcdFileAttributes, dtLastModified As Date,
dtLastAccessed As Date, dtCreated As Date, xMethod As xcdCompressionMethod,
bEncrypted As Boolean, sPassword As String, bExcluded As Boolean, xReason As
xcdSkippingReason, bExisting As Boolean)
```

Parameters

Parameter (Modifiable)	Description
sFilename (YES)	The file's path and filename, exactly as it will be stored in the zip file.
sComment (YES)	The file's comment.
sSourceFilename	The real path and filename of the file about to be zipped.
lSize	The file's size in bytes
xAttributes	The file's attributes
dtLastModified	The last modification date and time stamp.
dtLastAccessed	The last accessed date and time stamp.
dtCreated	The creation date and time stamp.
xMethod (YES)	The compression method
bEncrypted (YES)	Is this file going to be encrypted?
sPassword (YES)	If so, it will be encrypted with this password.
bExcluded (YES)	Will this file be zipped or excluded?

Applicable methods

Zip

UnzipPreprocessingFile event

Description

The UnzipPreprocessingFile event provides the opportunity to view and change a file's information before it is unzipped, and to make the final decision about whether the file should be unzipped or excluded from being unzipped. This event is triggered right before the output file is opened for unzipping.

Not all file information provided by the UnzipPreprocessingFile event's parameters is modifiable. The table below provides a list of parameters provided by UnzipPreprocessingFile and shows you which ones can be modified.

Declaration

(Parameters in bold are modifiable)

```
Event UnzipPreprocessingFile(sFilename As String, sComment As String, sDestFilename
As String, lSize As Long, lCompressedSize As Long, xAttributes As xcdFileAttibutes,
lCRC As Long, dtLastModified As Date, dtLastAccessed As Date, dtCreated As Date,
xMethod As xcdCompressionMethod, bEncrypted As Boolean, sPassword As String,
nDiskNumber As Long, bExcluded As Boolean, xReason As xcdSkippingReason, bExisting
As Boolean, xDestination As xcdUnzipDestination)
```

Parameters

Parameter (Modifiable)	Description
sFilename	The file's path and filename, exactly as it is currently stored in the zip file.
sComment	The file's comment.
sDestFilename (YES)	The real path and filename of the destination file about to be unzipped to.
lCompressedSize	The file's compressed size, in bytes
xAttributes	The file's attributes
lCRC	The file's uncompressed data CRC checksum.
dtLastModified	The last modification date and time stamp.
dtLastAccessed	The last accessed date and time stamp.
dtCreated	The creation date and time stamp.
xMethod	The compression method
bEncrypted	Is this file encrypted?
sPassword (YES)	If so, it will be decrypted with this password.
nDiskNumber	When dealing with spanned zip files, the disk this file is located on
bExcluded (YES)	Will this file be unzipped or excluded?
xReason	If the bExcluded flag (above) was True when the UnzipPreprocessingFile was triggered, this parameter contains the reason that the file was marked for excluding.
bExisting	Indicates whether the destination file about to be unzipped to already exists.
xDestination (YES)	Determines whether a file is unzipped to disk or to memory.

Applicable methods

Unzip

RemovingFile event

Description

The RemovingFile event is triggered once for each file that is about to be deleted from the zip file as a result of executing the RemoveFiles method. The RemovingFile event occurs just before the file is deleted.

None of the parameters provided by the RemovingFile event are modifiable.

Declaration

```
Event RemovingFile(sFilename As String, sComment As String, lSize As Long,
lCompressedSize As Long, xAttributes As xcdFileAttributes, lCRC As Long,
dtLastModified As Date, dtLastAccessed As Date, dtCreated As Date, xMethod As
XceedZipLibCtl.xcdCompressionMethod, bEncrypted As Boolean)
```

Parameters

Parameter	Description
sFilename	The path and filename of the file being deleted, exactly as it is stored in the zip file.
sComment	The file's comment.
lSize	The file's original uncompressed size in bytes.
lCompressedSize	The file's compressed size.
xAttributes	The file's attributes.
lCRC	The file's uncompressed data CRC checksum.
dtLastModified	The last modification date and time stamp.
dtLastAccessed	The last accessed date and time stamp.
dtCreated	The creation date and time stamp.
xMethod	The compression method
bEncrypted	Is this an encrypted file?

Applicable methods

RemoveFiles

TestingFile event

Description

The TestingFile event is triggered once for each file that has been verified as a result of calling the TestZipFile method. This event is triggered when verification is complete, and provides the integrity status and complete information on the verified file.

None of the parameters provided by the TestingFile event are modifiable.

Declaration

```
Event TestingFile(sFilename As String, sComment As String, lSize As Long,
lCompressedSize As Long, nCompressionRatio As Integer, xAttributes As
xcdFileAttributes, lCRC As Long, dtLastModified As Date, dtLastAccessed As Date,
dtCreated As Date, xMethod As xcdCompressionMethod, bEncrypted As Boolean,
nDiskNumber As Long)
```

Parameters

Parameter	Description
sFilename	The path and filename of the file being verified, exactly as it is stored in the zip file.
sComment	The file's comment.
lSize	The file's original uncompressed size in bytes.
lCompressedSize	The file's compressed size.
xAttributes	The file's attributes.
lCRC	The file's uncompressed data CRC checksum.
dtLastModified	The last modification date and time stamp.
dtLastAccessed	The last accessed date and time stamp.
dtCreated	The creation date and time stamp.
xMethod	The compression method
bEncrypted	Is this an encrypted file?
xIntegrity	The integrity of the file: Is it unzippable or is it corrupted?

Applicable methods

TestZipFile

FileStatus event

Description

The FileStatus event provides a status report on the processing of a given file. It is triggered during the processing of a file as a result of calling the Zip, Unzip or TestZipFile methods.

The FileStatus event is triggered for every 32K of uncompressed data that has been processed for the current file. If the file size is less than 32K, this event will only be triggered once. This event provides various statistics/information related to the processing of a file.

None of the parameters provided by the FileStatus event are modifiable.

Declaration

```
Event FileStatus(sFilename As String, lSize As Long, lCompressedSize As Long,
lBytesProcessed As Long, nBytesPercent As Integer, nCompressionRatio As Integer,
bFileCompleted As Boolean)
```

Parameters

Parameter	Description
sFilename	The path and filename of the file processed. When zipping, this is the path and filename of the file exactly as it is being stored in the zip file. When unzipping or testing, this is the path and filename of the file exactly as it is currently stored in the zip file.
lSize	The file's uncompressed size, in bytes
lCompressedSize	The file's compressed size, in bytes
lBytesProcessed	The amount of uncompressed bytes processed so far
nBytesPercent	The completion percentage of this file (from 0% to 100%) based on the amount of uncompressed bytes processed.
nCompressionRatio	When zipping, this is the compression ratio that has been achieved so far for this file. When unzipping, this is the compression ratio of the file as it is stored in the zip file.
bFileCompleted	This flag is set to True when the current file has been completely processed succesfully.

Remarks

When a file has been completely processed, the iBytesProcessed parameter equals the lSize parameter.

Applicable methods

Zip, Unzip, TestZipFile

Related topics

To get statistics on the entire group of files being processed, use the GlobalStatus event.

GlobalStatus event

Description

The GlobalStatus event provides a status report on the group of files currently being processed. It is triggered throughout the execution of the Zip, Unzip or TestZipFile methods.

The GlobalStatus event is triggered for every 32K of uncompressed data that has been processed for the entire group of files being processed. This event provides various statistics related to the group of files being processed.

None of the parameters provided by the GlobalStatus event are modifiable.

Declaration

```
Event GlobalStatus(lFilesTotal As Long, lFilesProcessed As Long, lFilesSkipped As
Long, nFilesPercent As Integer, lBytesTotal As Long, lBytesProcessed As Long,
lBytesSkipped As Long, nBytesPercent As Integer, lBytesOutput As Long,
nCompressionRatio As Integer)
```

Parameters

Parameter	Description
lFilesTotal	The total number of files to process
lFilesProcessed	The number of files that have been processed so far.
nFilesSkipped	The number of files that have been skipped so far.
nFilesPercent	The completion percentage (0% to 100%) based on how many files have been processed so far.
lBytesTotal	The total number of bytes to process for the entire group of files.
lBytesProcessed	The total number of bytes that have been processed so far for the entire group of files.
lBytesSkipped	The total number of bytes that have not been processed due to files that have been skipped.
lBytesPercent	The completion percentage (0% to 100%) based on the amount of bytes processed so far.
lBytesOutput	When unzipping, this represents the total number of bytes that have been output to the destination unzipping location. The value is undefined when zipping.
nCompressionRatio	When zipping, this is the compression ratio that has been achieved so far for the entire group of files being zipped. When unzipping, this is the compression ratio of the entire group of files being unzipped.

Remarks

When running the ListZipContents method, only the parameters that deal with files are available. The parameters that deal with bytes are not applicable.

Applicable methods

Zip, Unzip, TestZipFile

Related topics

To get statistics on individual files, as opposed to the entire group of files being processed, use the FileStatus event.

DiskNotEmpty event

Description

The DiskNotEmpty event is triggered when creating a spanned zip file and a non-empty disk is present or inserted into the removable media drive. This event is triggered only once per disk, right after the disk was requested by the InsertDisk event, but before writing any data to the disk.

The DiskNotEmpty event gives the host application a chance to perform various actions which might be necessary when a non-empty disk is present in the drive. The xNotEmptyAction parameter can be used when standard default actions to be taken (see below), or you may provide code for this event which performs custom actions. For example, you may want to create a separate "README.TXT" file or other data files on the disk before the actual zip file fills the disk space.

Declaration

```
Event DiskNotEmpty(xNotEmptyAction As xcdNotEmptyAction)
```

Parameters

Parameter (Modifiable)	Description
xNotEmptyAction (YES)	The action to take. Set it to xnaErase to erase the disk. Set it to xnaAppend to leave the files already on the disk alone, but fill up any remaining space with the zip file. Set it to xnaAskAnother to instruct the library to ask for another disk by triggering the InsertDisk event. Set it to xnaAbort to instruct the library to abort the current zipping operation.

Applicable methods

Zip

ProcessCompleted event

Description

The ProcessCompleted event is triggered whenever an Xceed Zip method has completed its processing. This event provides statistics about the completed operation similar to those provided by the GlobalStatus event.

The parameters provided by the ProcessCompleted event are not modifiable.

Declaration

```
Event ProcessCompleted(lFilesTotal As Long, lFilesProcessed As Long, lFilesSkipped
As Long, lBytesTotal As Long, lBytesProcessed As Long, lBytesSkipped As Long,
lBytesOutput As Long, nCompressionRatio As Integer, xResult As xcdError)
```

Parameters

Parameter	Description
lFilesTotal	The total number of files that were matched by the FilesToProcess property. For the ListZipContents and PreviewFiles mtehods, this is the only parameter among the first three listed here that has any meaning.
lFilesProcessed	The number of files that were actually processed. Does not include files that were skipped for any reason.
nFilesSkipped	The number of files that were skipped. Normally, lFilesProcessed + lFilesSkipped = lFilesTotal.
lBytesTotal	The total number of bytes for all the files that were matched by the FilesToProcess property.
lBytesProcessed	The total number of bytes that have were actually processed.
lBytesSkipped	The total number of bytes that have not been processed due to files that have been skipped. Normally, lBytesProcessed + lBytesSkipped = lBytesTotal.
lBytesOutput	When zipping, this represents the total number of bytes that have been written to the zip file. When unzipping, this represents the total number of bytes that have been output to the destination unzipping location.
nCompressionRatio	When zipping, this is the compression ratio that has been achieved so far for the entire group of files being zipped. When unzipping, this is the compression ratio of the entire group of files being unzipped.
xResult	The result code returned by the method.

Remarks

In your handler for the ProcessCompleted event, the CurrentOperation property allows you to find out which method just completed.

Applicable methods

All methods except the AddFilesToProcess and AddFilesToExclude methods.

ZipComment event

Description

The ZipComment event is triggered right before the first file is processed as a result of a call to the Zip, Unzip or ListZipContents methods. The event provides the Zip file's "zip comment" - a string or data block that can be up to 64K of uncompressed data. The zip comment is usually useful for providing a short comment about the entire zip file.

When zipping, the sComment parameter provided by this event can be modified in order to change or add a zip file comment.

Declaration

(Parameters in bold are modifiable)

```
Event ZipComment(sComment As String)
```

Parameters

Parameter (Modifiable)	Description
sComment (YES)	The zip file's comment (modifiable when running the Zip method)

Applicable methods

Zip, Unzip or ListZipContents

QueryMemoryFile event

Description

The QueryMemoryFile event is triggered when the Zip method is called, and allows you to zip up data directly from memory into a file entry in the zip file. You can cause the QueryMemoryFile event to be triggered multiple times if you have more than one file you wish to zip up directly from memory.

Each time the QueryMemoryFile event is triggered, you must provide a filename to give the data you are zipping up from memory, and you can also specify file attributes, a file comment, and other file information for the data.

Set the QueryMemoryFile event's bFileProvided flag to true if you do indeed have a file to zip up directly from memory. Whenever this flag is set to True, the QueryMemoryFile event will be triggered again, so you can provide another filename to zip up. Set the flag to False when you do not have any more files to zip up directly from memory.

If you don't write a handler for the QueryMemoryFileEvent, no files will be zipped directly from memory because the default value for the bFileProvided flag is False.

The actual data for each file to zip up from memory must be provided later, when the ZippingMemoryFile event is triggered.

Declaration

(Parameters in bold are modifiable)

```
Event QueryMemoryFile(lUserTag As Long, sFilename As String, sComment As String,
xAttributes As xcdFileAttributes, dtLastModified As Date, dtLastAccessed As Date,
dtCreated As Date, bEncrypted As Boolean, sPassword As String, bFileProvided As
Boolean)
```

Parameters

Parameter (Modifiable)	Description
lUserTag (YES)	A user defined tag you can specify. This tag is provided to you later, when the ZippingMemoryFile event is triggered. If you don't explicitly set this tag to your own value, it starts at 0 and is incremented when the next QueryMemoryFile event is triggered.
sFilename (YES)	The file to create in the zip file which will contain the data to be compressed directly from memory.
sComment (YES)	A comment for the file.
xAttributes (YES)	The file's attributes.
dtLastModified (YES)	The file's last modification date and time.
dtLastAccessed (YES)	The file's last accessed date and time.
dtCreated (YES)	The file's creation date and time.
bEncrypted (YES)	Should this file be encrypted?
sPassword (YES)	Use this password to encrypt the file.
bFileProvided (YES)	Flag informing Xceed Zip that you have a file to zip up from memory.

Applicable methods

Zip

ZippingMemoryFile event

Description

The ZippingMemoryFile event is triggered so that you can provide the data of a file that is being zipped from memory. The ZippingMemoryFile event is triggered at least once for each file that you have specified with the QueryMemoryFile event.

Declaration

(Parameters in bold are modifiable)

```
Event ZippingMemoryFile(lUserTag As Long, sFilename As String, vaDataToCompress As Variant, bEndOfData As Boolean)
```

Parameters

Parameter (Modifiable)	Description
lUserTag	This is the value of the lUserTag parameter you specified when the QueryMemoryFile event was triggered.
sFilename	This is the value of the sFilename parameter you specified when the QueryMemoryFile event was triggered.
baDataToCompress (YES)	This is where you provide the data to zip up.
bEndOfData (YES)	Set this to True if the entire data to compress is provided to the baDataToCompress parameter. When dealing with streaming data, or if you aren't able to provide the entire data for the current file being zipped up from memory, set this parameter to False. This will cause the ZippingMemoryFile event to be triggered again (with the same lUserTag and sFilename parameters) so that you can provide more data. Set bEndOfData to True when you have finished providing data to zip up from memory for the current file.

Applicable methods

Zip

UnzippingMemoryFile event

Description

The UnzippingMemoryFile event is triggered to provide you with the data of a file that is being unzipped to memory. The UnzippingMemoryFile event is only triggered if you have specified xudMemory or xudMemoryStream for the xDestination parameter in the UnzipPreprocessingFile event.

The parameters provided by the UnzippingMemoryFile event are not modifiable.

Declaration

```
Event UnzippingMemoryFile(sFilename As String, vaUncompressedData As Variant,
bEndOfData As Boolean)
```

Parameters

Parameter	Description
sFilename	This is the filename for the file who'se data is currently being provided to you.
baData	This is the data of the file.
bEndOfData	When this flag is False, it indicates to you that there is still more data to be provided for the current file. In this case, the UnzippingMemoryFile event will be triggered again (with the same sFilename parameter) so you can get more data. When this flag is True, there is no more data to provide for the current file. This flag is always True for files that had the UnzipPreprocessingFile event's xUnzippingDestination parameter set to xudMemory.

Applicable methods

Unzip

About Xceed Zip control properties

The following Xceed Zip control properties are used by the various Xceed Zip methods in order to determine their behavior.

Most of these properties contain default values that are appropriate for common tasks, and can be altered visually at design-time using your language's property editor window. At run time, your code can change the values of the properties as desired before calling an Xceed Zip method. Some properties only apply to a few Xceed Zip methods. The "Applicable methods" section in the documentation for each property lists the methods that refer to the property.

List of the Xceed Zip control's properties:

- Abort
- BasePath
- BackgroundProcessing
- CompressionLevel
- CurrentOperation
- EncryptionPassword
- ExcludedFileAttributes
- ExtraHeaders
- FilesToProcess
- FilesToExclude
- MinDateToProcess
- MaxDateToProcess
- MinSizeToProcess
- MaxSizeToProcess
- PreservePaths
- ProcessSubfolders
- RequiredFileAttributes
- SfxBinaryModule
- SkipIfExisting
- SkipIfNotExisting
- SkipIfOlderDate
- SkipIfOlderVersion
- SpanMultipleDisks
- SplitSize
- TempFolder
- UseTempFile
- UnzipToFolder
- ZipFilename
- ZipOpenedFiles

About Xceed Zip control methods

The Xceed Zip control provides the following methods to accomplish the tasks of compressing, uncompressing, and manipulating zip files. Each method's behavior is determined by the values of various properties. The "Applicable properties" section in the documentation for each method lists the properties that are used by that method. When executing a method, it is important to correctly set all properties that are used by that method. Unwanted side effects may occur if properties were altered for the execution of one method, but were not set to the desired values for the next execution of a method that uses those same properties.

Most methods provide a return value that indicates if the operation was successful or if a warning or error has occurred. A properly coded application always checks, at the very least, to make sure the operation was successful. See the Error codes topic for a list of possible return values for all methods that return an xcdError type result code.

List of the Xceed Zip control's methods:

- Zip
- Unzip
- ListZipContents
- GetZipFileInformation
- Convert
- PreviewFiles
- RemoveFiles
- TestZipFile
- AddFilesToProcess
- AddFilesToExclude
- GetErrorDescription
- License

About Xceed Zip control events

Events generated by the Xceed Zip control represent activities that your program can recognize and act upon. For example, the Zipping event occurs when a file is being zipped. So if you want to, you can write an event handler for the Zipping event that displays "Currently zipping file:" on a status bar along with the filename and any other file info you want for the file currently being zipped.

With **Visual FoxPro 5.0** and up, events will be triggered only if you set Application.AutoYield = .F.. Place this code in the section of your main program where you set up your environment. If you only have a single form in your application that uses Xceed Zip, place Application.AutoYield = .F. in the forms load method, and set it back to the default in the forms destroy method.

The following events are generated by the Xceed Zip control. The "Applicable methods" section in each event's description topic indicates which methods trigger the event during their execution.

Code in event handlers for Xceed Zip events cannot recursively call Xceed Zip methods.

List of the Xceed Zip control's events:

- DiskNotEmpty
- FileStatus
- GlobalStatus
- InsertDisk
- InvalidPassword
- ListingFile
- PreviewingFile
- ProcessCompleted
- QueryMemoryFile
- RemovingFile
- SkippingFile
- TestingFile
- UnzippingMemoryFile
- UnzipPreprocessingFile
- Warning
- ZipComment
- ZipContentsStatus
- ZippingMemoryFile
- ZipPreprocessingFile

About Xceed Zip control enumeration types

The following enumerations are built into the Xceed Zip control's ActiveX interface and are used as return codes for Xceed Zip methods, as parameter types for Xceed Zip methods and events, and as types for some Xceed Zip properties.

List of global constants:

xcdLastDisk = 0

xcdMinimumDate = 01/01/1900

xcdMaximumDate = 12/31/9999

List of the Xceed Zip control's enumeration types:

- xcdCompressionLevel
- xcdCompressionMethod
- xcdCurrentOperation
- xcdDiskSpanning
- xcdError
- xcdFileAttributes
- xcdNotEmptyAction
- xcdSkippingReason
- xcdUnzipDestination
- xcdValueType
- xcdWarning

About Xceed Zip control error codes

The following constants define the possible return values of all the Xceed Zip methods of type xcdError, as well as the xResult parameter of the ProcessCompleted event. The declarations for these error codes are built into the ActiveX control's interface and should automatically be accessible by your code.

The sample applications include a function that displays a detailed message depending on the return value provided. You can copy this function from the sample application directly into your own project(s) if desired. It is useful for debugging purposes, or if you want your application to display messages if an error occurs.

Notable return values of type xcdError:

- xerSuccess (0)
- xerProcessStarted (1)
- xerWarnings (526)
- xerFilesSkipped (527)

All other error codes of type xcdError:

- xerEmptyZipFile (500)
- xerSeekInZipFile (501)
- xerEndOfZipFile (502)
- xerOpenZipFile (503)
- xerCreateTempFile (504)
- xerReadZipFile (505)
- xerWriteTempZipFile (506)
- xerWriteZipFile (507)
- xerMoveTempFile (508)
- xerNothingToDo (509)
- xerCannotUpdateAndSpan (510)
- xerMemory (511)
- xerSplitSizeTooSmall (512)
- xerSFXBinaryNotFound (513)
- xerReadSFXBinary (514)
- xerCannotUpdateSpanned (515)
- xerBusy (516)
- xerInsertDiskAbort (517)
- xerUserAbort (518)
- xerNotAZipFile (519)
- xerUninitializedString (520)
- xerUninitializedArray (521)
- xerInvalidArrayDimensions (522)
- xerInvalidArrayType (523)
- xerCannotAccessArray (524)
- xerUnsupportedDataType (525)
- xerDiskNotEmptyAbort (528)
- xerRemoveWithoutTemp (529)
- xerNotLicensed (530)
- xerInvalidSfxProperty (531)
- xerInternalError (999)

About Xceed Compression control methods

The Xceed Compression control provides the following methods for compressing and decompressing data entirely in memory. Each method's behavior is determined by the values of the Xceed Compression control's properties. The "Applicable properties" section in the documentation for each method lists the properties that are used by that method. Mainly, you will want to consult the properties if you are encrypting your data, or if you want to change the compression factor.

The methods listed below provide a return value that indicates if the operation was successful or if a warning or error has occurred. A properly coded application always checks, at the very least, to make sure the operation was successful. See the Error codes topic for a list of possible return values for these methods.

List of the Xceed Compression control's methods:

- Compress
- Uncompress
- CalculateCrc
- GetErrorDescription
- License

About Xceed Compression control properties

The following Xceed Compression control properties are used by the memory compression methods in order to determine their behavior.

These properties contain default values that are appropriate for common tasks, and can be altered visually at design-time using your language's property editor window. At run time, your code can change the values of the properties as desired before calling an Xceed Compression method. Some properties only apply to a few Xceed Compression methods. The "Applicable methods" section in the documentation for each property lists the methods that refer to the property.

List of the Xceed Compression control's properties:

- CompressionLevel
- EncryptionPassword

About Xceed Compression enumeration types

The following enumerations are built into the Xceed Compression control's ActiveX interface and are used as return codes for Xceed Compression methods.

List of the Xceed Zip control's enumeration types:

- xcdCompressionError
- xcdCompressionLevel

About Xceed Compression control error codes

The following constants define the possible return values of all the Xceed Compression control's methods of type xcdCompressionError. The declarations for these error codes are built into the ActiveX control's interface and should automatically be accessible by your code.

Error codes returned:

- xceSuccess (0)
- xceSessionOpened (1000)
- xceInitCompression (1001)
- xceCompression (1003)
- xceInvalidPassword (1004)
- xceChecksumFailed (1005)
- xceDataRemaining (1006)

Spanned zip files

A spanned zip file is a zip file that is split into multiple files, each located on a different floppy disk or other removable media. Spanned zip files are particularly useful when there is a need to create a zip file that is too large to fit on a single floppy disk or other removable media.

Zip method example for VB

The following code uses the Zip method to compress the file "C:\TEST\MY FILE.DAT" into the zip file "C:\TEST\MY TEST.ZIP". You should check the Zip method's return value once you are done – if it is xerSuccess, the operation was successful. Otherwise, look up the error code in the error codes topic to find out what went wrong. The following code assumes you placed a button and an Xceed Zip control on a form and named them Command1 and XceedZip1 respectively.

```
Sub Command1_Click()
    Dim ResultCode As xcdError
    ' All properties keep their default values except the two below
    XceedZip1.FilesToProcess = "c:\test\my file.dat"
    XceedZip1.ZipFilename = "c:\test\my test.zip"
    ' Start zipping
    ResultCode = XceedZip1.Zip
    ' Check the return value.
    If ResultCode <> xerSuccess Then
        MsgBox "Unsuccessful. Error # " + Str(nErr) + " occurred. " +
            "Description: " + XceedZip1.GetErrorDescription(xvtError,
    ResultCode)
    Else
        MsgBox "File(s) succesfully zipped."
    End If
End Sub
```

Zip method example for Delphi

The following code uses the Zip method to compress the file "C:\TEST\MY FILE.DAT" into the zip file "C:\TEST\MY TEST.ZIP". You should check the Zip method's return value once you are done – if it is xerSuccess, the operation was successful. Otherwise, look up the error code in the error codes topic to find out what went wrong. The following code assumes you placed a button and an Xceed Zip control on a form and named them Button1 and XceedZip1 respectively.

```
procedure TForm1.Button1Click(Sender: TObject);
var
    ResultCode: xcdError;
begin
    { All properties keep their default values except the two below }
    XceedZip1.FilesToProcess :=  'c:\test\my file.dat';
    XceedZip1.ZipFilename := 'c:\test\my test.zip';
    { Start zipping }
    ResultCode := XceedZip1.Zip;
    { Check the return value. If ResultCode = xerSuccess, it worked,
      otherwise, display a message box that with a description of
      the error. Use the GetErrorDescription function. }
end;
```

Filtering properties

Some Xceed Zip properties are designed for the sole purpose of allowing you to control what types of files are processed among those specified in the FilesToProcess property. The filtering properties are:

- FilesToExclude
- MinDateToProcess
- MaxDateToProcess
- MinSizeToProcess
- MaxSizeToProcess
- RequiredFileAttributes
- ExcludedFileAttributes

xcdError error code enumeration

```
xcdCurrentOperation
{
	xerSuccess = 0,
	xerProcessStarted = 1,
	xerEmptyZipFile = 500,
	xerSeekInZipFile = 501,
	xerEndOfZipFile = 502,
	xerOpenZipFile = 503,
	xerCreateTempFile = 504,
	xerReadZipFile = 505,
	xerWriteTempZipFile = 506,
	xerWriteZipFile = 507,
	xerMoveTempFile = 508,
	xerNothingToDo = 509,
	xerCannotUpdateAndSpan = 510,
	xerMemory = 511,
	xerSplitSizeTooSmall = 512,
	xerSFXBinaryNotFound = 513,
	xerReadSFXBinary = 514,
	xerCannotUpdateSpanned = 515,
	xerBusy = 516,
	xerInsertDiskAbort = 517,
	xerUserAbort = 518,
	xerNotAZipFile = 519,
	xerUninitializedString = 520,
	xerUninitializedArray = 521,
	xerInvalidArrayDimensions = 522,
	xerInvalidArrayType = 523,
	xerCannotAccessArray = 524,
	xerUnsupportedDataType = 525,
	xerWarnings = 526,
	xerFilesSkipped = 527,
	xerDiskNotEmptyAbort = 528,
	xerRemoveWithoutTemp = 529,
	xerNotLicensed = 530,
	xerInvalidSfxProperty = 531,
	xerInternalError = 999
}
```

This enumeration is used by most all the Xceed Zip control methods that return a result code. Click on any of the above parameters to display an explanation of their meaning.

Relative paths

A relative path is a path that must be combined with a root path in order to make a complete (absolute) path. For example, "GRAPHICS\256COLOR\SMALL" is a relative path because it does not start by indicating a device or drive. "C:\USERS\FRED\GRAPHICS\256COLOR\SMALL" is an absolute path.

Current directory

The current directory is the folder from which an application was executed. An application's current directory can change depending on where it is installed or by which method is chosen to start it (it could be executed from a batch file, by double-clicking on its program icon, from a shortcut, etc.)

It is recommended that you do not rely on the applications current directory when specifying paths and files to process or exclude. Use the BasePath property instead.

Controlling how paths are stored in the zip file

Each file stored in a zip file must have a filename associated with it. A zip file can also contain path information for each file.

When zipping up files using the Zip method, you may or may not want path information stored in the zip file, or you may need to store only a portion of a path. Xceed Zip provides you with total control over how filenames and paths are stored in the zip file. The different techniques you can use are described below.

Use the FilesToProcess property

In order to use the zip method, you must use the FilesToProcess property to specify the files you want to zip up. As mentioned in the FilesToProcess property topic, you can specify files to zip up either with absolute paths, relative paths or without paths at all.

- If you specify files using absolute paths, the entire path to the file, except the drive or device portion, will be stored in the zip file. For example, if you specify "C:\DATA*.*" or "\\COMPUTERNAME\SHARENAME\DATA*.*" in FilesToProcess, files will be stored in the zip file without the "C:\" or "\\COMPUTERNAME\SHARENAME\" portion of the path.
- If you specify files using relative paths, the paths are assumed to be relative to the application's current directory, unless you use the BasePath property, which lets you specify the path files are relative to. In both cases, only the relative portion of a path is stored in the zip file.
- If you specify filenames without paths, the files will be searched for in the current directory, or the directory specified in the BasePath property if it is used. When specifying only filenames, no path information will be stored in the zip file.

Use the ZipPreprocessingFile event

Immediately before each file is actually zipped up into the zip file, the ZipPreprocessingFile event is triggered. This event provides you with the sFilename parameter, which contains the path and filename of the file about to be zipped. Modifying this parameter allows you to store any path and filename you want in the zip file.

Use the PreservePaths property

Setting the PreservePaths property to False will cause all path information to be discarded, so only the filename will be stored in the zip file.

Absolute paths

An absolute path is a complete path. It can start by a drive letter and colon or it can be a UNC format path. Examples of absolute paths are "C:\DATA FILES", "D:\" or "\\COMPUTERNAME\SHARENAME\DATA FILES". A path such as "GRAPHICS\256COLOR\SMALL" is not an absolute path. It is a relative path because the root portion is missing.

Brief introduction to the zip file format

A zip file is a single file that contains one or more files. Zip files usually have the ".ZIP" filename extension, which clearly identifies them.

Each file contained in a zip file is either stored as compressed data or as raw uncompressed data, and can be encrypted or not encrypted. Together with a file's data and filename, a zip file can also store each file's name, attributes and size. It can also store a path for each file, as well as the the file's date and time stamps, a user-defined comment and a CRC checksum of the file's uncompressed data.

The zip file format is extensible, so other information about a file can be stored in it while still remaining compatible with existing unzipping applications. For example, the new version 4.0 of the Xceed Zip Compression Library can store and retrieve Windows NT file attributes and permissions, the file's UNICODE filename and other file information not usually found in zip files.

Structurally, a zip file consists of one or more sections, each containing a file header followed by a file's compressed or raw uncompressed data. The file headers contain information such as the file's filename, file attributes and size. Following these sections is a compact "central directory" section, which contains (mostly redundant) information about each file stored in the zip file. This central directory is very useful for quickly determining what the contents of the zip file are, because it eliminates the need to scan through the entire zip file in order to find the file headers. At the very end of the zip file, an optional "zip comment" section can be provided and can be as large as 64K.

The zip format also supports spanned zip files. For spanned zip files, the central directory section is located on the last disk, unless it is a very large zip file, in which case the central directory may be located on the last two or more disks. Reading the central directory is usually the first step an unzipping application takes in order to unzip a file, which explains why the last disk is usually requested first when unzipping files.

The Xceed Zip Compression Library also allows you to create spanned zip files that are not located on different removable-media disks. Because these parts of the zip file are located in the same place, they must have different filenames (the first part of the zip file is named ".ZIP", the second one is named ".Z01", the third is named ".Z03" and so on.) It is important to note that most currently available unzipping applications (except those that use Xceed Zip v4.0 and up as their zip/unzip engine) cannot unzip spanned zip files that are in the same location until each part of the zip file is placed on separate removable media disks and renamed to ".ZIP".

CRC

Cyclic Redundancy Check. The CRC is a 32-bit checksum system that is used to verify the integrity of data. The zip file format contains the 32-bit CRC of each file's original uncompressed data. When a file is unzipped, the CRC is calculated on the file's newly unzipped data and compared with the stored CRC of the original file. If the CRCs are the same, then the unzipped file is very probably identical to the original file, because the 32-bit CRC is able to detect single bit variations even in the largest files.

BasePath property example

Let's say you want to zip all the .BMP files in the "C:\SERVER5\COMMON\DATA\BITMAPS\" folder, but you only want the "DATA\BITMAPS\" portion of the path to be stored before each filename. Note the following code:

```
XceedZip1.ZipFilename = "C:\TEMP\TEST.ZIP"
XceedZip1.FilesToProcess = "C:\SERVER5\COMMON\DATA\BITMAPS\*.BMP"
XceedZip1.PreservePaths = True
ResultCode = XceedZip1.Zip
```

If you use the above code, your files will be stored in the zip file with the following (incorrect) path and filenames (the drive letter and colon are always discarded):

```
"SERVER5\COMMON\DATA\BITMAPS\PICTURE1.BMP"
"SERVER5\COMMON\DATA\BITMAPS\PICTURE2.BMP"
…
```

To get the desired results, use the following code:

```
XceedZip1.ZipFilename = "C:\TEMP\TEST.ZIP"
XceedZip1.BasePath = "C:\SERVER5\COMMON"
XceedZip1.FilesToProcess = "DATA\BITMAPS\*.BMP"
XceedZip1.PreservePaths = True
ResultCode = XceedZip1.Zip
```

The files will then be stored with the following (correct) path and filenames instead:

```
"DATA\BITMAPS\PICTURE1.BMP"
"DATA\BITMAPS\PICTURE2.BMP"
…
```

xcdCompressionLevel enumeration

```
xcdCompressionLevel
{
      xclNone = 0,
      xclLow = 1,
      xclMedium = 6,
      xclHigh = 9
}
```

This enumeration is used by the CompressionLevel property. Explanations on each parameter can be found in that property's help topic.

SkippingFile event

Description

The SkippingFile event is triggered whenever a file is excluded from processing or cannot be processed at all. It is also triggered whenever the TestZipFile method encounters a corrupted file.

The SkippingFile event provides complete information on the file being skipped, as well as the reason it is being skipped. The information provided by the SkippingFile event can be displayed on the screen, put into a log file, printed out, or completely ignored – it's your choice. If you do not write a handler for the SkippingFile event, your application will still be notified if any files are skipped because the method currently executing will return a result code of either xerFilesSkipped or xerNothingToDo (if all files to be processed end up being skipped).

The parameters provided by the SkippingFile event are not modifiable.

Declaration

```
Event SkippingFile(sFilename As String, sComment As String, sFilenameOnDisk As
String, lSize As Long, lCompressedSize As Long, xAttributes As xcdFileAttributes,
lCRC As Long, dtLastModified As Date, dtLastAccessed As Date, dtCreated As Date,
xMethod As xcdCompressionMethod, bEncrypted As Boolean, xReason As
xcdSkippingReason)
```

Parameters

Parameter	Description
sFilename	The path and filename of the file being skipped. When zipping files, this is the path and filename of the file exactly as it would have been stored in the zip file if it was zipped. When unzipping files, this is the path and filename of the file exactly as it is currently stored in the zip file.
sComment	The file's comment.
sFileOnDisk	The real path and filename of the file being skipped. When zipping files, this is the path and filename of the actual file being zipped. When unzipping files, this is the path and filename that would have been given to the file if it was unzipped.
lSize	The file's original uncompressed size in bytes.
lCompressedSize	The file's compressed size.
xAttributes	The file's attributes.
lCRC	The file's uncompressed data CRC checksum.
dtLastModified	The last modification date and time stamp.
dtLastAccessed	The last accessed date and time stamp.
dtCreated	The creation date and time stamp.
xMethod	The compression method
bEncrypted	Is this an encrypted file?
xReason	If it will be excluded, why? Values for the xReason parameter that range between 1 and 99 are skipping reasons generated for files that were skipped due to the settings of the filtering properties or the SkipIf* properties. Values of 100 or more are skipping reasons generated because of an error while processing the file.

Remarks

Most of the sample applications included with Xceed Zip include a handler for the SkippingFile event which displays a message box indicating which file is being skipped and the reason for skipping the file. You can copy the SkippingFile event handler from the sample application directly into your own application if desired, so you won't have to write any code to handle this event.

RequiredFileAttributes property example

If you only want to process files that have the "Archive bit" on, and skip all other files, then you must set the RequiredFileAttributes property to xfaArchive.

If you only want to process files that have the "Hidden" or "System" attributes on, and skip all other files, then you must set the RequiredFileAttributes property to xfaHidden+xfaSystem.

But…

If you don't want to process files that have the "Archive bit" on, but you do want to process all other files, then you need to use the ExcludedFileAttributes property, not the RequiredFileAttributes property.

ExcludedFileAttributes property example

If you don't want to process files that have the "Archive bit" on, but you do want to process other files, then you must set the ExcludedFileAttributes property to xfaArchive.

If you don't want to process files that have the "Hidden" or "System" attributes on, but you do want to process other files, then you must set the ExcludedFileAttributes property to xfaHidden+xfaSystem.

But…

If you only want to zip files that have the "Archive bit" on, and skip all other files, then you need to use the RequiredFileAttributes property, not the ExcludedFileAttributes property.

Wildcards

Xceed Zip supports the * and ? wildcard characters. The * character matches a sequence of 0 or more characters. The ? character matches exactly one character.

Do not use "*.*" to match all files - you must use "*" instead. In the 32-bit Windows world, using *.* will only match files that contain a "." in them!

Use the ProcessSubfolders property to determine whether or not the contents of encountered subdirectories are also processed.

Wildcard matching is not case sensitive.

FilesToExclude property example

To zip up all the .TXT files in the "C:\DATA" folder and the "D:\EXTRAS" folder, except those .TXT files that start with "TMP", set the FilesToProcess and FilesToExclude properties to the following values:

```
FilesToProcess = "C:\DATA\*.TXT" + CHR$(13) + "D:\EXTRAS\*.TXT"
FilesToExclude = "TMP*"
```

MinDateToProcess and MaxDateToProcess properties example

To only zip or unzip files that have a date and time stamp in the month of January 2000, set the MinDateToProcess and MaxDateToProcess to January 1, 2000 12:00am and January 31, 2000 11:59pm respectively.

MinSizeToProcess and MaxSizeToProcess properties example

To only zip or unzip files that are bigger than 1024 bytes but not larger than 1048576 bytes, set the MinSizeToProcess and MaxSizeToProcess properties to 1024 and 1048576 respectively.

SkipIf* properties

The Skip* group of properties includes: SkipIfExisting, SkipIfNotExisting, SkipIfOlderDate and SkipIfOlderVersion. These properties let you control whether files should be processed or not based on properties of files that already exist in the destination zip file or unzipping location.

xcdDiskSpanning enumeration

```
xcdDiskSpanning
{
      xdsNever = 0,
      xdsAlways = 1,
      xdsAutoDetect = 2
}
```

This enumeration is used by SpanMultipleDisks property. Explanations on each parameter can be found in that property's help topic.

xcdCurrentOperation enumeration

```
xcdCurrentOperation
{
      xcoIdle = 0,
      xcoPreviewing = 1,
      xcoListing = 2,
      xcoZipping = 3,
      xcoUnzipping = 4,
      xcoRemoving = 5,
      xcoTestingZipFile = 6,
      xcoGettingInformation = 7,
      xcoConverting = 8
}
```

This enumeration is used by CurrentOperation property. Explanations on each parameter can be found in that property's help topic.

xcdExtraHeader enumeration

```
xcdExtraHeader
{
      xehNone = 0,
      xehExtTimeStamp = 1,
      xehFileTimes = 2,
      xehSecurityDescriptor = 4,
      xehUnicode = 8
}
```

This enumeration is used by the ExtraHeaders property. Explanations on each parameter can be found in the that property's help topic.

Convert method

Description

This method converts one type of zip file into another. While converting, any problems in the zip file are automatically corrected when possible, therefore this method can also be used to repair zip files. You can perform the following types of conversions and repairs:

- Convert a regular zip file into a self-extracting zip file
- Convert a self-extracting zip file into a regular zip file
- Convert a self-extracting zip file into a new self-extracting zip file with a different configuration
- Repair a zip file

The source zip file or self-extracting zip file must be specified in the ZipFilename property. The destination zip file or self-extracting zip file must be specified with the sDestFilename parameter. If the destination is intended to be a self-extracting zip file, you must specify a self-extractor binary in the SfxBinaryModule property. If the destination is intended to be a regular zip file, you must leave the SfxBinaryModule property empty.

Declaration

```
Method Convert(sDestFilename As String) As xcdError
```

Parameters

Parameter	Description
sDestFilename	The path and filename of the converted zip file. (Specify a ".EXE" file extension and a binary in the SfxBinaryModule property if you want to convert the zip file to a self-extracting zip file)

Return values

A return value of type xcdError is returned. Only the return value of xerSuccess indicates that the method has been completed succesfully. See the error codes topic for a list of possible return values.

Remarks

None

Applicable properties

Abort, ZipFilename, ExtraHeaders, SfxBinaryModule and the Sfx* properties.

Events triggered

ProcessCompleted

Related topics

If you want to create self-extracting zip file without requiring an already existing zip file, use the Zip method in conjunction with the SfxBinaryModule property.

SfxBinaryModule property

Description

The SfxBinary property is used to inform the library to create self-extracting zip files instead of regular zip files, and to provide the filename and location of the binary to use for creating the self-extracting Zip files.

You can use the binaries provided with the Xceed Zip Self-Extractor Module (XCDSFX16.BIN, XCDSFX32.BIN) or any 3rd party binary such as the Info-Zip groups free SFX binary (contact Xceed Software, we'll send it to you). Only the binaries provided with the Xceed Zip Self-Extractor Module are configurable.

If you do not specify a path and filename in the SfxBinaryModule property, the Zip method will create regular zip file.

Data type

String

Default value

Empty (Create regular zip files, not self-extracting zip files)

Applicable methods

Zip, Convert

Related topics

None

SfxButtons property

Description

The SfxButtons property contains the captions for all the buttons that are shown by self-extracting zip files created with the Xceed Zip Self-Extractor Module. Button captions can contain the & character, which becomes invisible and causes the following character to be underlined and considered a windows keyboard shortcut character to the button (also known as a keyboard accelerator).

As an indexed string property, each entry in the array is a caption for a different button.

The maximum length string allowed for each entry is 30 characters.

Data type

xcdSfxButtons

Property array indexes

Value	Meaning
xsbOK	This is the caption for the OK buttons. The default value is "&OK".
xsbCancel	This is the caption for the Cancel button, which is displayed in the folder browser, and can only be changed in the 16-bit self-extractor. The 32-bit self-extractor uses the system's browser and cannot be customized by Xceed Zip. The default value is "&Cancel".
xsbAbort	This is the caption for the Abort button that is displayed on the Progress dialog during extraction. The button allows the user to stop the self-extracting process. The default value is "&Abort".
xsbSkip	This is the caption for the button that provides the option to skip a file. The button is available whenever the Password Prompt dialog is displayed. The default value is "&Skip File".
xsbAlwaysSkip	This is the caption for the button that provides the option to always skip files that cannot be decrypted, without any further prompting with the password dialog. The button is available in the password dialog. The default value is "&Always Skip".
xsbYes	This is the caption for Yes buttons. The default value is "&Yes".
xsbNo	This is the caption for No buttons. The default value is "&No".
xsbOverwriteAll	This is the caption for the button that provides the option to always overwrite files that already exist, without any further prompting with the Overwrite dialog. The button is available in the Overwrite dialog. The default value is "&All".
xsbOverwriteNone	This is the caption for the button that provides the option to never overwrite files that already exist, without any further prompting with the Overwrite dialog. The button is available in the Overwrite dialog. The default value is "N&one".
xsbContinue	This is the caption for the button that is displayed in the introduction message box, and provides the option to continue, as oposed to exiting if the user clicks on the exit button. The default value is "&Continue".
xsbExit	This is the caption for the button that is displayed in the introduction message box and the Select Unzipping Folder dialog. It provides the option to exit the self-extractor without extracting any files. The default value is "&Exit".
xsbAgree	This is the caption for the Agree button that is displayed in the License agreement dialog box. This button usually indicates that the user accepts the terms of the license agreement and can proceed with the self-extraction process. The default value is "&Agree".
xsbRefuse	This is the caption for the Refuse button that is displayed in the License agreement dialog box. This button usually indicates that the user does not accept the terms of the license agreement and can not proceed with the self-extraction process. The default value is "&Refuse".

Default values

The default value for each button is listed in the above table.

Browsable

No

Remarks

The values in this property are consulted right before a call to the Zip method (when creating a self-extracting zip file) and the Convert method.

Applicable methods

Zip, Convert

Related topics

None

SfxMessages property

Description

The SfxMessages property contains the text that is displayed by the various information dialog boxes shown by the self-extractor. The values in this property are consulted by Xceed Zip when you are building a self-extracting Zip file.

As an indexed string property, each entry in the array is text used in a different information dialog box.

For 32-bit components, the maximum length string allowed for each entry in this property is 2000 characters, except for the license agreement message which can contain up to 8000 characters. For 16-bit components, the maximum length for all SfxMessages string entries are 255 characters.

If you do not want the self-extractor to display any messages or dialogs, use the SfxClearMessages method to disable them.

Data type

xcdSfxMessages

Property array indexes

Value	Meaning
xsmSuccess	This message is displayed in the Success dialog box upon successful extraction of all the files in the archive. The default value is "All files were successfully unzipped."
xsmFail	This message is displayed in the Fail dialog box if an error occurs during the extraction process. The following macros are available when specifying a value for the fail message: %e = The error code that caused the failure. The default value is "An error occurred while unzipping. One or more files were not successfully unzipped. The error code is %e."
xsmErrorCreatingFolder	This message is displayed in the Error Creating Folder dialog box. The following macros are available when specifying a value for ErrorCreatingDirMsg: %d = The folder that could not be created. The default value is "Unable to create folder '%d'."
xsmIntro	This message is displayed in the Introduction dialog box. The default value is "Welcome to the Xceed Zip Self-Extractor. This program will unzip some files onto your system."
xsmLicense	This is the message displayed in the License agreement dialog box. The license agreement dialog box allows you to display a license agreement and to force the user to either accept or reject the terms of the agreement. If the terms are rejected, the self-extractor will terminate and no files will be extracted.
xsmDestinationFolder	This prompt is displayed in the Select Unzip Folder dialog box. The prompt asks the user to select the path where the files will be extracted to. The default value is "Select the folder where you want to unzip the files to."
xsmPassword	This prompt asks the user to enter a password for a given file. It is displayed in the Password Prompt dialog box. The default value is "A password is required for file:"
xsmInsertLastDisk	This prompt asks the user to insert the last disk of the set of disks containing the self-extractor and its files. This prompt is used when the self-extractor spans multiple disks, and the last disk is required. The default value is "This self-extracting zip file is part of a multidisk zip file. Please insert the last disk of the set."
xsmInsertDisk	This prompt asks the user to insert a specific disk from the set of disks containing the self-extractor and its files. This prompt is used when the self-extractor spans multiple disks, and a specific disk is required.

	The following macros are available when specifying a value for InsertDiskPrompt: %n = The number of the requested disk. The default value is "Please insert disk #%n."
xsmAbortUnzip	This prompt asks the user if they really wish to abort the self-extracting process, after having clicked on the "Abort" button (see AbortBtn). The default value is "Are you sure you want to abort the unzipping process?"
xsmCreateFolder	This is the text displayed in the Create Folder dialog. The text informs the user that the folder where files will be unzipped to does not exist, and asks the user if they wish to create the folder. The following macros are available when specifying a value for CreateFolderPrompt: %d = The folder to create The default value is "Folder '%d' does not exist. Do you want to create it?"
xsmOverwrite	This prompt asks the user if they wish to overwrite a file that already exists. It is used in the Overwrite dialog. See the Overwrite setting for information on what choices the user has when the Overwrite dialog is displayed. The following macros are available when specifying a value for OverwritePrompt: %f = The path and filename of the file to overwrite The default value is "The file '%f' already exists. Do you want to overwrite it?"
xsmProgess	This is the string used to indicate which file is currently being extracted. It is used in the Progress dialog box. The default value is "Unzipping file:"

Default value

The default value for each message is listed in the above table.

Browsable

No

Applicable methods

Zip, Convert

Related topics

None

SfxStrings property

Description

The SfxStrings property contains label captions and other miscellaneous text that are found inside the various dialog boxes displayed by the self-extractor. The values in this property are consulted by Xceed Zip when you are building a self-extracting Zip file.

As an indexed string property, each entry in the array is the text of a different label.

The maximum length string allowed for each entry in this property is 255 characters, except for the TitleStr (index 2) entry, which can only be 100 characters long.

Data type

xcdSfxStrings

Property array indexes

Value	Meaning
xssProgressBar	This is the string used to explain the purpose of the progress gauge that shows the overall progress of the unzip operation. The default value is "Overall progress:"
xssTitle	This is string that appears in the title bar of every message box and dialog box displayed by the Xceed Zip self-extractor. The default value is "The Xceed Zip Self-Extractor"
xssCurrentFolder	This is the string that is used in the folder browser to indicate the user-selected folder where the files will be unzipped to. The default value is "Current destination folder:"
xssShareName	This is the string used to display the available connected drives in the folder browser's drive list. The following macros are available when specifying a value for DriveShareStr: %s = Share name %c = Computer name where share is located The default value is "%s on '%c'"
xssNetwork	This is the string used to represent the entry in the folder browser's drive list that represents the entire network. The default value is "Entire Network"

Default value

The default value for each string is listed in the above table.

Browsable

No

Applicable methods

Zip, Convert

Related topics

None

SfxDefaultPassword property

Description

The SfxDefaultPassword property lets you specify the password that the self-extracting zip file will use whenever it tries to unzip an encrypted file. If this property is left empty, or the password specified is incorrect for any of the encrypted files, then the self-extractor's *Password Prompt* dialog will be displayed (if it has not been disabled by clearing the password prompt entry in the SfxMessages property).

This property can be useful for developers that want to create self-extracting zip files that cannot be unzipped by programs such as WinZip™ without the appropriate decryption password.

The maximum length string allowed for this property is 80 characters.

Data type

String

Default value

Empty

Browsable

Yes

Applicable methods

Zip, Convert

SfxDefaultUnzipToFolder property

Description

The SfxDefaultUnzipToFolder property lets you specify the path where the self-extractor extracts files to by default. This can be changed later by the user if the SfxMessages property has a non-empty value set for xsmDestinationFolder. If the SfxDefaultUnzipToFolder property is empty, the self-extracting zip file's current directory at run-time will be used.

The following macros are available when specifying a value for the SfxDefaultUnzipToFolder property: %w, %s,%t, %r, %e, %p and %v.

The maximum length string allowed for this property is 255 characters.

Data type

String

Default value

Empty

Browsable

Yes

Applicable methods

Zip, Convert

Related topics

None

SfxExistingFileBehavior property

Description

The SfxOverwrite property lets you determine the overwrite behavior exhibited by self-extracting zip files whenever files are being extracted that already exist in the destination folder.

Possible values

Value	Meaning
xseAsk	Prompt user to confirm with the Overwrite dialog. The Overwrite dialog prompts the user on how to proceed. The user can select the "Yes" button (see YesBtn) to overwrite the file, the "No" button (see NoBtn) to skip the file, the "All" button (see OverwriteAllBtn) to overwrite the file and any subsequent files that already exist, and finally the "None" button (see OverwriteNoneBtn) to skip the file and any other files that already exist.
xseSkip	Never overwrite files, and do not prompt user.
xseOverwrite	Always overwrite files, and do not prompt user.

Data type

xcdSfxExistingFileBehavior

Default value

xseAsk

Browsable

Yes

Applicable methods

Zip, Convert

Related topics

None

SfxReadmeFile property

Description

The SfxReadmeFile property lets you specify the path and filename of a text file that will be displayed by the self-extractor upon successfully extracting its files. The file is displayed using Notepad, right before the executable specified by the SfxExecuteAfter property is run.

If SfxReadmeFile is empty, no text file will be displayed.

The maximum length string allowed for this property is 255 characters.

Data type

String

Default value

Empty

Browsable

Yes

Applicable methods

Zip, Convert

SfxExecuteAfter property

Description

The SfxRunExePath property lets you specify the path and filename of an executable that the self-extractor will run upon successfully extracting its files.

A command line for the executable can be specified by adding a space after the path and filename, followed by the command line.

The following macros are available when specifying a value for the SfxRunExePath property: %w, %s, %d, %t, %e, %p and %v.

If the SfxRunExePath property is empty, no program will be run.

The maximum length string allowed for this property is 255 characters.

Data type

String

Default value

Empty

Browsable

Yes

Applicable methods

Zip, Convert

Related topics

None

SfxInstallMode property

Description

The SfxInstallMode property is a special mode you can put the self-extractor in. When SfxInstallMode is set to True, the self-extractor will wait until the application executed by the self-extractor (see the SfxRunExePath property) has terminated, and then delete all the files that were extracted and any new directories that were created. When SfxInstallMode is True and nothing is specified in the SfxDefaultUnzipToFolder property (recommended), the self-extractor will extract files in a random folder in the systems temporary folder (the equivalent of putting %t\%r in the SfxDefaultUnzipToFolder property). Also, the self-extractors Select Unzip Folder dialog box is not displayed when SfxInstallMode is True.

This property is particularly useful when the self-extractor is being used in order to extract and run an install or setup program. Once the setup program has finished copying all the files to the appropriate locations, and has terminated, setting the SfxInstallMode property to True allows you to ensure that the now unneeded files are erased from the system.

Data type

Boolean

Default value

False

Browsable

Yes

Applicable methods

Zip, Convert

Related topics

None

SfxProgramGroup property

Description

The SfxProgramGroup property lets you specify the name of a program group that the self-extractor will create after successfully unzipping files. If this property is left empty, a program group will not be created.

Use the SfxProgramGroupItems property to specify the contents of the program group created by the self-extractor.

The program group is created before the text file specified in the SfxReadmeFile property is displayed, and before the executable specified in the SfxExecuteAfter property is executed.

Data type

String

Default value

Empty

Browsable

Yes

Applicable methods

Zip, Convert

Related topics

None

SfxProgramGroupItems property

Description

The SfxProgramGroupItems property lets you specify the items to be created in the program group specified in the SfxProgramGroup property.

Each item in the list consists of a path+filename as well as a name. You must separate the file+pathname from the name using the | character (ascii value 124). Each item must be separated by a CR character (ascii value 13) or the CR/LF characters (ascii value 13 followed by ascii value 10). See below for an example.

The SfxAddProgramGroupItem method allows you to easily add items to this property's list of program group items.

The following macros are available when specifying a value for the path+filename portion of an item: %w, %s, %t, %e, %d, %p and %v.

Here is an example of a string used to provide 3 items to the SfxProgramGroupItems property using VB:

```
XceedZip1.SfxProgramGroupItems = "%d\Tetris\Tetris.exe|Play Tetris Now" +
    chr$(13) + "%d\Tetris\readme.txt|Read me!" + chr$(13) + "%d\Tetris\
    Register.exe|Register Online!"
```

In Delphi, you would do this:

```
XceedZip1.SfxProgramGroupItems.Add('%d\Tetris\Tetris.exe|Play Tetris Now');
XceedZip1.SfxProgramGroupItems.Add('%d\Tetris\readme.txt|Read me!');
XceedZip1.SfxProgramGroupItems.Add('%d\Tetris\Register.exe|Register Online!');
```

Data type

String

Default value

Empty

Browsable

Yes

Applicable methods

Zip, Convert

Related topics

None

SfxExtensionsToAssociate property

Description

The SfxExtensionsToAssociate property lets you assign filename extensions that the self-extractor will register with Windows.

Registering a filename extension allows Windows to recognize files that have a specific filename extension. Windows will be able to show a description of the files ending with a given extension (in Windows Explorer, for example) and execute an application whenever that type of file is opened. Thus, when specifying each filename extension to be registered, you must provide a description and an associated application. The associated application must be able to open the files ending with that extension.

The entire list of extensions to register are placed into one string that you assign to the SfxExtensionsToAssociate property. Each extension to register consists of the extensions description, the extension (do not include the . character) and the path+filename of the associated application. You must separate the description, extension and file+pathname by using the | character (ascii value 124). Each extension to register must be separated by a CR character (ascii value 13) or the CR/LF characters (ascii value 13 followed by ascii value 10). See below for an example.

The SfxAddExtensionToAssociate method allows you to easily add items to this property's list of extensions to associate.

The following macros are available when specifying a value for the path+filename of the application associated to each extension: %w, %s, %d, %t, %e, %p and %v.

Here is an example of a string used to provide 3 extensions to the SfxExtensionsToAssociate property using VB:

```
XceedZip1.SfxRegisterExtensions = "Zip file|zip|%d\ZipView\ZipView.exe" +
    chr$(13) + "High score file|sco|%p\Tetris\Tetris.exe" + chr$(13) + "Read
    me file|me|%w\notepad.exe"
```

In Delphi, you would do this:

```
XceedZip1.SfxRegisterExtensions := 'Zip file|zip|%d\ZipView\ZipView.exe' + #13
    + 'High score file|sco|%p\Tetris\Tetris.exe' + #13 + 'Read me file|me|%w\
    notepad.exe';
```

The extensions are registered right after the program group specified by the SfxProgramGroup property is created, and before the text file specified in the SfxReadmePath property is displayed and the executable specified in the SfxRunExePath is executed.

Data type

String

Default value

Empty

Browsable

Yes

Applicable methods

Zip, Convert

Related topics

None

SfxIconFilename property

Description

The SfxIconFilename property lets you customize the self-extracting zip file's application icon. This is the icon Windows uses to display the executable's icon, as well as the icon that appears in the title bar of all the self-extractors dialog boxes.

You can assign the filename and path of any valid windows icon (.ICO) file to this property. When this property is left empty, no icon is assigned and the self-extracting zip file will use its built-in "zip" icon.

Data type

String

Default value

Empty

Browsable

Yes

Applicable methods

Zip, Convert

Related topics

None

GetZipFileInformation method

VB example Delphi example

Description

The GetZipFileInformation method allows you to obtain some general statistics about the zip file specified in the ZipFilename property.

The method returns the zip file information by modifying the parameters you pass to the function – the contents of the parameters are never checked by the GetZipFileInformation method.

Declaration

```
Method GetZipFileInformation(lNbFiles As Long, lCompressedSize As Long,
lUncompressedSize As Long, nCompressionRatio As Integer, bSpanned As Boolean) As
xcdError
```

Parameters

Parameter	Description
lNbFiles	Total number of files contained in the zip file
lCompressedSize	Total compressed size of all the data in the zip file
lUncompressedSize	Total uncompressed size of the data in the zip file
nCompressionRatio	The compression ratio achieved by this zip file
bSpanned	Returns true if the zip file spans multiple disks

Return values

A return value of type xcdError is returned. Only the return value of xerSuccess indicates that the method has been completed succesfully. See the error codes topic for a list of possible return values.

Remarks

If the zip file spans multiple disks, the InsertDisk event will be triggered in order to request the last disk of the set.

Applicable properties

Abort, ZipFilename

Events triggered

InsertDisk, Warning

Related topics

Use the ListZipContents method to get information on the files contained in the zip file, as opposed to general statistics on the zip file.

SfxAddProgramGroupItem method

Description

The SfxAddProgramGroupItem method adds an item to the SfxProgramGroupItems property's list of items to insert in the program group.

SfxAddProgramGroupItem automatically handles separating items with the CR+LF characters. In fact, it performs the same work as the following line of code:

```
XceedZip1.SfxProgramGroupItems = XceedZip1.SfxProgramGroupItems + sApplication
    + "|" + sDescription + Chr(13) + Chr(10)
```

Declaration

```
Method SfxAddProgramGroupItem(sApplication As String, sDescription As String)
```

Parameters

Parameter	Description
sApplication	The filename related to the item being added
sDescription	A short description of the item

Return values

None

Applicable properties

SfxProgramGroupItems

SfxAddExtensionToAssociate method

Description

The SfxAddExtensionToAssociate method adds an item to the SfxExtensionsToAssociate list of extensions to associate.

SfxAddExtensionToAssociate automatically handles separating items with the CR+LF characters. In fact, it performs the same work as the following line of code:

```
XceedZip1.SfxExtensionsToAssociate = XceedZip1.SfxExtensionsToAssociate +
    sDescription + "|" + sExtension + "|" + sApplication + Chr(13) + Chr(10)
```

Declaration

```
Method SfxAddExtensionToAssociate(sDescription As String, sExtension As String,
sApplication As String)
```

Parameters

Parameter	Description
sDescription	A short description of the item
sExtension	The extension to associate to the application
sApplication	The path and filename of the item that will be associated to

Return values

None

Applicable properties

SfxExtensionsToAssocitate

SfxResetButtons method

Description

The SfxResetButtons method resets all the entries in the SfxButtons property back to their original defaults.

Declaration

```
Method SfxResetButtons()
```

Parameters

None

Return values

None

Applicable properties

SfxButtons

SfxResetStrings method

Description

The SfxResetStrings method resets all the entries in the SfxStrings property back to their original defaults.

Declaration

```
Method SfxResetStrings()
```

Parameters

None

Return values

None

Applicable properties

SfxStrings

SfxResetMessages method

Description

The SfxResetMessages method resets all the entries in the SfxMessages property back to their original defaults.

Declaration

```
Method SfxResetMessages()
```

Parameters

None

Return values

None

Applicable properties

SfxMessages

SfxClearButtons method

Description

The SfxClearButtons method clears all the entries in the SfxButtons property. This will save a small amount of space in the configuration portion of a self-extracting zip file. Usually, the SfxClearButtons method is used in conjunction with the SfxClearMessages method.

Declaration

```
Method SfxClearButtons()
```

Parameters

None

Return values

None

Applicable properties

SfxButtons

SfxClearMessages method

Description

The SfxClearMessages method clears all the entries in the SfxButtons property. If all the entries are empty in the SfxButtons property at the time the Zip or ConvertToSelfExtracting methods are called, then the resulting self-extracting zip file will never display any dialog or message boxes on screen, nor prompt the user for passwords, new disks or file replace conditions. In other words, if you want completely silent operation, call SfxClearMessages method.

Declaration

```
Method SfxClearMessages()
```

Parameters

None

Return values

None

Remarks

If the purpose of a call to the SfxClearMessages method is to achieve completely silent operation (no windows displayed by the self-extracting zip file), then you should also call the SfxClearButtons and SfxClearStrings methods to erase all prompts, strings and button captions from the configuration portion of the self-extracting zip file – you will obtain slightly smaller self-extracting zip files this way.

Applicable properties

SfxButtons

SfxClearStrings method

Description

The SfxClearStrings method clears all the entries in the SfxStrings property. This will save a small amount of space in the configuration portion of a self-extracting zip file. Usually, the SfxClearStrings method is used in conjunction with the SfxClearMessages method.

Declaration

```
Method SfxClearStrings()
```

Parameters

None

Return values

None

Applicable properties

SfxStrings

Warning event

Description

The Warning event is generated whenever a recoverable problem is encountered during the execution of one of the Xceed Zip methods. The event provides you with the filename of the file that was being processed when the problem was detected, and the warning code indicating the nature of the problem.

Declaration

```
Event Warning(sFilename As String, xWarning As xcdWarning)
```

Parameters

Parameter	Description
sFilename	This is the filename of the file currently being processed
xWarning	This is a code indicating the type of problem detected with the file.

Remarks

If you do not write a handler for this event, you will still get a return value of xerWarnings when the method you are calling completes. This will provide you with an indication that there were warnings despite not having written any code for this event.

InvalidPassword event

Description

The InvalidPassword event is triggered the first time an encrypted file is encountered when unzipping or testing files. The event allows you to supply a decryption password to be used in order to decrypt the file and therefore be able to unzip it. The InvalidPassword event is also triggered whenever files are encountered that cannot be decrypted with the password supplied the last time the InvalidPassword event was triggered.

When unzipping files, if no handler is written for the InvalidPassword event, encrypted files will not be unzipped or tested. If an incorrect password is supplied, the InvalidPassword event will be triggered again until either the correct password is supplied, or the command is issued to skip the current file being unzipped or tested.

Note that the SkippingFile event occurs for any encrypted files that are skipped due to an invalid password or if Xceed Zip is instructed to skip the current file being unzipped.

Declaration

(Parameters in bold are modifiable)

```
Event InvalidPassword(sFilename As String, sNewPassword As String, bRetry As
Boolean)
```

Parameters

Parameter	Description
sFilename	The filename of the file that requires the password in order to be decrypted.
sNewPassword	Provide the encryption password here.
bRetry	Set this to True if you have provided a new password. Leaving this parameter to False (the default value it has when the event is triggered) will cause Xceed Zip to skip the file from being unzipped.

xcdFileAttributes enumeration

```
xcdFileAttributes
{
      xfaNone = 0,
      xfaReadOnly = 1,
      xfaHidden = 2,
      xfaSystem = 4,
      xfaVolume = 8,
      xfaFolder = 16,
      xfaArchive = 32
}
```

This enumeration is used by various Xceed Zip control events. It is the type for the xAttributes parameter. It is also used by the RequiredFileAttributes and ExcludedFileAttributes properties. Explanations for each of the values belonging to the xcdFileAttributes enumeration can be found in each property or event's topic description.

xcdCompressionMethod enumeration

```
xcdCompressionMethod
{
      xcmStored = 0,
      xcmDeflated = 8
}
```

This enumeration is used by the ListingFile, UnzipPreprocessingFile, ZipPreprocessingFile, RemovingFile, TestingFile and SkippingFile events. Explanations on each parameter can be found in the corresponding event's help topic.

xcdNotEmptyAction enumeration

```
xcdNotEmptyAction
{
      xnaErase = 0,
      xnaAppend = 1,
      xnaAskAnother = 2,
      xnaAbort = 3
}
```

This enumeration is used by the InsertDisk event. Explanations on each parameter can be found in the appropriate event's help topic.

xcdSkippingReason enumeration

```
xcdSkippingReason
{
	xsrIncluded = 0,
	xsrFilesToExclude = 1,
	xsrSkipExisting = 2,
	xsrSkipNotExisting = 3,
	xsrSkipOlderDate = 4,
	xsrSkipOlderVersion = 5,
	xsrRequiredAttributes = 6,
	xsrExcludedAttributes = 7,
	xsrMinDate = 8,
	xsrMaxDate = 9,
	xsrMinSize = 10,
	xsrMaxSize = 11,
	xsrSkipUser = 12,
	xsrDuplicateFilenames = 13,
	xsrSkipReplace = 14,
	xsrUpdateWithoutTemp  = 115,
	xsrInvalidDiskNumber = 100,
	xsrFolderWithSize = 101,
	xsrWriteFile = 102,
	xsrOpenFile = 103,
	xsrReadFile = 104,
	xerMoveFile = 105,
	xsrInvalidPassword = 106,
	xsrInvalidCRC = 107,
	xsrInvalidUncompSize = 108,
	xsrCentralHeaderData = 109,
	xsrLocalHeaderData = 110,
	xsrDescriptorHeaderData = 111,
	xsrCreateFolder = 112,
	xsrAccessDenied = 113,
	xsrCreateFile = 114,
	xsrUpdateWithoutTemp = 115,
	xsrRenamedToExisting = 116,
	xsrVolumeWithSize = 117,
	xsrCannotSetVolumeLabel = 118,
	xsrCentralHeaderNotFound = 119,
	xsrUnzipDiskFull = 120,
	xsrCompress = 121,
	xsrUncompress = 122
}
```

This enumeration is used by SkippingFile event. Explanations on each parameter can be found in that event's help topic.

xcdSfxButtons enumeration

```
xcdSFXButtons
{
      xsbOk = 0,
      xsbCancel = 1,
      xsbAbort = 2,
      xsbSkip = 3,
      xsbAlwaysSkip = 4,
      xsbYes = 5,
      xsbNo = 6,
      xsbOverwriteAll = 7,
      xsbOverwriteNone = 8,
      xsbContinue = 9,
      xsbExit = 10,
      xsbAgree = 11,
      xsbRefuse = 12
}
```

This enumeration is used by the SfxButtons property. Explanations on each parameter can be found in the that property's help topic.

xcdSfxMessages enumeration

```
xcdSfxMessages
{
      xsmSuccess = 0,
      xsmFail = 1,
      xsmErrorCreatingFolder = 2,
      xsmIntro = 3,
      xsmLicense = 4,
      xsmDestinationFolder = 5,
      xsmPassword = 6,
      xsmInsertLastDisk = 7,
      xsmInsertDisk = 8,
      xsmAbortUnzip = 9,
      xsmCreateFolder = 10,
      xsmOverwrite = 11,
      xsmProgress = 12
}
```

This enumeration is used by the SfxMessages property. Explanations on each parameter can be found in the that property's help topic.

xcdSfxStrings enumeration

```
xcdSfxStrings
{
      xssProgressBar = 0,
      xssTitle = 1,
      xssCurrentFolder = 2,
      xssShareName = 3,
      xssNetwork = 4
}
```

This enumeration is used by the SfxStrings property. Explanations on each parameter can be found in the that property's help topic.

xcdSfxExistingFileBehavior enumeration

```
xcdSfxExistingFileBehavior
{
      xseAsk = 0,
      xseSkip = 1,
      xseOverwrite = 2
}
```

This enumeration is used by the SfxExistingFileBehavior property. Explanations on each parameter can be found in the that property's help topic.

xerBusy constant

Value

516

Description

The current instance of the library is already running another operation.

Tips

Although the library is fully multi-threaded, each instance of the Xceed Zip control can only do one operation at a time. This makes event handling much clearer. The developer can create another instance of the library if it is necessary to do another, independent zipping or unzipping operation simultaneously.

xerCannotAccessArray constant

Value

524

Description

The data in the string or array passed to the vaDataToCompress parameter of the ZippingMemoryFile event cannot be accessed by the library.

Tips

Make sure the variant or variable is a valid 1-dimensional array of byte values, or a string variable.

xerCannotUpdateAndSpan constant

Value

510

Description

The library was instructed to update a zip file that ended up being too big to fit in the current location. A zip file cannot be updated from a regular zip into a spanned or split zip file.

Tips

The spanned or split zip file must be completely recreated in order to be updated into a spanned or a split zip file.

xerCannotUpdateSpanned constant

Value

515

Description

The library was instructed to remove or add files to an existing spanned or split zip file. This type of operation is not supported.

Tips

The spanned or split zip file must be recreated without the files you want to remove, or recreated with the entire group of files that you want it to contain.

xerCreateTempFile constant

Value

504

Description

An error occurred when trying to create a temporary file.

Tips

Make sure that the path you have specified in the TempFolder property is valid and not protected by security or access permissions your application does not have. Temporary files are opened for exclusive writing.

xerEmptyZipFile constant

Value

500

Description

The zip file is empty.

Tips

The zip file contains no files, although it still has the structure of a zip file (including an empty central directory)

xcdWarning enumeration

```
xcdWarning
{
	xwrIncompleteWrite = 300,
	xwrInvalidSignature = 301,
	xwrInvalidCentralOffset = 302,
	xwrInvalidLocalOffset = 303,
	xwrInvalidDescriptorOffset = 304,
	xwrJunkInZip = 305,
	xwrSecurityNotSupported = 306,
	xwrSecurityGet = 307,
	xwrSecuritySet = 308,
	xwrSecuritySize = 309,
	xwrSecurityVersion = 310,
	xwrSecurityUnknownCompression = 311,
	xwrSecurityData = 312,
	xwrUnicodeSize = 313,
	xwrUnicodeData = 314,
	xwrExtraHeaderSize = 315,
	xwrTimeStampSize = 316,
	xwrTimeStampFlags = 317,
	xwrFileTimesSize = 318,
	xwrInvalidFileCount = 319
}
```

This enumeration is used by the Warning event. Click on any of the above parameters to display an explanation of their meaning.

xerEndOfZipFile constant

Value

502

Description

Premature end-of-file encountered in the zip file.

Tips

This can happen when the library is reading data from the zip file, but an end-of-file condition occurs. The zip file is probably truncated, or is corrupted.

xerInsertDiskAbort constant

Value

517

Description

The library was instructed to abort the current operation during the occurrence of the InsertDisk event.

Tips

This can happen when the bDiskInserted parameter of the InsertDisk event is left to its default value of False. The library takes this as an indication that no handler was written for the InsertDisk event. Of course, this error will also occur if the parameter is voluntarily set to False.

xerInternalError constant

Value

999

Description

An unexpected error occurred in the library.

Tips

There are some types of errors tested in the library's code that should not occur in practice. If you get this error value, please contact Xceed Software immediately.

xerInvalidArrayDimensions constant

Value

522

Description

The array passed in the vaDataToCompress parameter of the ZippingMemoryFile event was not a 1-dimensional array.

Tips

Make sure the variant or variable indeed a valid 1-dimensional byte array or a string variable.

xerInvalidArrayType constant

Value

523

Description

The array passed in the vaDataToCompress parameter of the ZippingMemoryFile event was not a byte array.

Tips

Make sure the variant or variable is a valid 1-dimensional array of byte values, or a string variable.

xerMemory constant

Value

511

Description

The system ran out of memory.

Tips

A memory allocation requested by the library was refused by the operating system. The machine was low on available memory.

xerMoveTempFile constant

Value

508

Description

Cannot move the temporary zip file to the real zip file destination specified by the ZipFilename property.

Tips

When zipping, and the UseTempFile property is set to True, the library creates the zip file in a temporary file. When it has completed the operation, it attempts to move the temporary file to the destination zip file. A copy is made if the source and destination are on different drives. This error can happen when the application using the library does not have the permissions required to delete the destination zip file, even though the library was able to open it for writing before the operation started as a check to avoid this type of error...

xerNotAZipFile constant

Value

519

Description

The zip file specified in the ZipFilename property is invalid.

Tips

The library has determined that the file is not a zip file, because a required signature was not found anywhere in the file.

xerNothingToDo constant

Value

509

Description

No files ended up being processed.

Tips

This can happen when all the files specified to the FilesToProcess property ended up being skipped (keep in mind that the SkippingFile event is triggered for each file that's skipped) due to reasons such as being unable to find or open the file, the file was excluded by the FilesToExclude or one of the other filtering properties, etc. This can also happen if you try to delete files from an already empty zip file, or from a non-existent zip file.

xerSuccess constant

Value

0

Description

This value indicates that the method's process executed successfully, without errors, warnings, or skipped files.

xerOpenZipFile constant

Value

503

Description

The zip file could not be opened or found.

Tips

The path and filename provided to the ZipFilename property may be invalid. When trying to zip or update files in a zip file, this could mean that the zip file is already opened for reading or writing by another application. When unzipping files or listing the zip file's contents, this could mean the zip file is already opened by another application for writing.

xerReadSfxBinary constant

Value

514

Description

An error occurred while reading data from the self-extractor binary specified in the SfxBinaryModule property.

Tips

This can happen if the network drive where the self-extractor binary is located on suddenly becomes disconnected, or if the floppy disk the self-extractor binary is located on is removed from the drive while it is being read by the library. The library was able to open the self-extractor binary for reading, though.

xerReadZipFile constant

Value

505

Description

An error occurred while reading data from the zip file.

Tips

This can happen if the network drive where the zip file is located on suddenly becomes disconnected, or if the floppy disk the zip file is located on is removed from the drive while it is being unzipped from the library. It can also happen if there is a bad sector on the drive where the zip file is located.

xerSeekInZipFile constant

Value

501

Description

Cannot seek in the zip file.

Tips

This can happen when the library is looking for signatures that identify the zip file but is unable to perform a seek operation. Make sure you are using a random-access device.

xerSfxBinaryNotFound constant

Value

513

Description

Could not find the self-extractor binary specified in the SfxBinaryModule property.

Tips

Verify the path and filename of the specified self-extractor binary. This error can also happen if the library cannot open the file for read operations. It may be opened in an exclusive mode by another application, or the application using the library may not have the permissions required to open the file.

xerSplitSizeTooSmall constant

Value

512

Description

The value specified in the SplitSize property was too small.

Tips

The split size is smaller than the biggest of the zip file header's. The headers themselves cannot span file boundaries – they must reside on the same disk/split file. In any case, do not attempt to split zip files smaller than 1024 bytes – this is highly inefficient for most filesystems.

xerUninitializedArray constant

Value

521

Description

An invalid byte array was passed in the vaDataToCompress parameter of the ZippingMemoryFile event.

Tips

Make sure the variant or variable passed is indeed a valid byte array.

xerUninitializedString constant

Value

520

Description

An invalid string was passed in the vaDataToCompress parameter of the ZippingMemoryFile event.

Tips

Make sure the variant or variable passed is indeed a string or byte array.

xerUnsupportedDataType constant

Value

525

Description

An invalid data type was received by the library for the vaDataToCompress parameter of the ZippingMemoryFile event.

Tips

Only byte arrays or strings can be used.

xerUserAbort constant

Value

518

Description

The abort property was set to True, and the current operation was succesfully aborted.

xerWriteTempZipFile constant

Value

506

Description

An error occurred while writing to a temporary file.

Tips

Make sure that the drive where your system's temporary folder resides has enough disk space and is accessible, or the drive pointed to by the TempFolder property has enough disk space and is accessible.

xerWriteZipFile constant

Value

507

Description

An error occurred while writing to the zip file.

Tips

Make sure that the drive where your zip file is being written to is not full and has not been disconnected or rendered inaccessible. This error can happen if the network drive the zip file is located on suddenly becomes disconnected, or if the floppy disk the zip file is located on is removed from the drive while it is being created or updated by the library.

About Xceed Zip control warnings

The following constants define the possible values of the Warning event's xWarning parameter.

The declarations for these constants are built into the ActiveX control's interface and should automatically be accessible by your code.

Possible values for the xWarning parameter:

- xwrIncompleteWrite (300)
- xwrInvalidSignature (301)
- xwrInvalidCentralOffset (302)
- xwrInvalidLocalOffset (303)
- xwrInvalidDescriptorOffset (304)
- xwrJunkInZip (305)
- xwrSecurityNotSupported (306)
- xwrSecurityGet (307)
- xwrSecuritySet (308)
- xwrSecuritySize (309)
- xwrSecurityVersion (310)
- xwrSecurityUnknownCompression (311)
- xwrSecurityData (312)
- xwrUnicodeSize (313)
- xwrUnicodeData (314)
- xwrExtraHeaderSize (315)
- xwrTimeStampSize (316)
- xwrTimeStampFlags (317)
- xwrFileTimesSize (318)
- xwrInvalidFileCount (319)

xwrExtraHeaderSize constant

Value

315

Description

The total size of all the extra headers in the zip file exceeds the 64K boundary allowed by the format.

Tips

The zip file is corrupted.

xwrFileTimesSize constant

Value

318

Description

Could not restore a file's "filetime" header information from the zip file because the extra header containing this information has an invalid size.

Tips

The time stamp information will not be restored for this file. This header contains the three dates (creation time, last modification time and last accessed time) for the file. The file's last modification time will still be restored because if is contained in the zip file's central directory entry for each file.

xwrIncompleteWrite constant

Value

300

Description

A write operation to the zip file failed. The component was attempting to dump buffered data to the zip file before closing it.

Tips

The resulting zip file, if present, may be corrupted. The zip file should be verified and recreated if necessary.

xwrInvalidCentralOffset constant

Value

302

Description

The zip file's central directory has an entry with an incorrect offset. This can happen when unexpected data is found between two central directory entries, or when the first entry in the central directory is located past the offset indicated in the zip file's end-of-central-directory header.

Tips

The zip file is damaged.

xwrInvalidDescriptorOffset constant

Value

304

Description

Unexpected data was found between a compressed file's data and the file's descriptor header. A file's descriptor header is normally located after the file's compressed data at an offset determined by information provided by the file's local header. In this case, the descriptor header was found past the expected offset.

Tips

The zip file is damaged.

xwrInvalidLocalOffset constant

Value

303

Description

A file's local header entry in the zip file was found past the offset reported in the zip file's central directory.

Tips

The zip file is damaged.

xwrInvalidSignature constant

Value

301

Description

A signature that identifies a zip file was not found, or was the incorrect signature. The file may not be a zip file.

Tips

Make sure it is indeed a zip file by calling the GetZipFileInformation method.

xwrJunkInZip constant

Value

305

Description

An error occurred while compressing a file, but the library was unable to remove the file's data that was already compressed and written into the zip file. This can happen, for example, when a file being compressed to floppy disks has already spanned a disk and then becomes inaccessible to the library. The library would normally seek back into the zip file, erase the already written compressed data for the file, and continue with the next file to zip. It cannot do this if a new disk was inserted.

Tips

The zip file will contain extra data that will not affect other files, but that makes the zip file unnecessarily larger. It is therefore recommended to rezip the files or pass the zip file through the convert method, to eliminate the extra data that was inserted due to this situation.

xwrSecurityData constant

Value

312

Description

Could not restore a file's security information from the zip file because the security descriptor's data is corrupted.

Tips

In this case the library will not restore a file's security information, but the file will be unzipped anyway.

xwrSecurityGet constant

Value

307

Description

Could not obtain a file's security information. That means the library was unable to store a file's security information in the the zip file as requested by the setting of the ExtraHeaders property.

Tips

Either the filesystem does not support security, or the operating system reports that it cannot provide the security information to the library. The latter case is probably occuring because the user that's running the software using the library does not have the privileges to obtain a file's security information.

xwrSecurityNotSupported constant

Value

306

Description

The library was unable to write a security descriptor header, or to recreate a file's security information. When zipping, this can happen when you have requested that the library write a security descriptor header (see the ExtraHeaders property) but one of the files being zipped is located on a filesystem that does not support security permissions. When unzipping, this can happen when you have requested that the library restore security information stored in the zip file, and the destination filesystem or folder does not support security.

Tips

When this warning occurs, you know a file's security information has not been stored or restored. You can choose to either zip or unzip files from or to a filesystem that does support security, or remove the xehSecurity constant from the ExtraHeaders property.

xwrSecuritySet constant

Value

308

Description

Could not restore a file's security information from the zip file. That means the library was unable to write a file's security information to disk, as requested by the setting of the ExtraHeaders property.

Tips

Either the filesystem does not support security, or the operating system reports that it cannot provide the security information to the library. The latter case is probably occuring because the user that's running the software using the library does not have the privileges to obtain a file's security information.

xwrSecurityUnknownCompression constant

Value

311

Description

Could not restore a file's security information from the zip file because the security descriptor was compressed using an unknown compression method.

Tips

The library supports security descriptors compressed with the deflate algorithm. It is conceivable, although unlikely, that there are some libraries or applications that may attempt to compress the security descriptor with another compression algorithm that was used in previous versions of the zip file format. If this is the case, library will unzip the file anyway, but without restoring its security information. If you come across a zip file that produces this warning, we suggest you contact Xceed Software and let us know.

xwrSecurityVersion constant

Value

310

Description

Could not restore a file's security information from the zip file because the security descriptor version is not recognized.

Tips

The zip file was created with an application that creates security extra headers that are not currently supported by the Xceed Zip Compression Library. The file will be unzipped anyway, but without restoring its security information. If you come across a zip file that produces this warning, we suggest you contact Xceed Software and let us know.

xwrTimeStampFlags constant

Value

317

Description

Could not restore a file's "timestamp" header information from the zip file because the extra header containing this information contains invalid information.

Tips

The time stamp information will not be restored for this file. This header contains the three dates (creation time, last modification time and last accessed time) for the file. The file's last modification time will still be restored because if is contained in the zip file's central directory entry for each file.

xwrTimeStampSize constant

Value

316

Description

Could not restore a file's "timestamp" header information from the zip file because the extra header containing this information has an invalid size.

Tips

The time stamp information will not be restored for this file. This header contains the three dates (creation time, last modification time and last accessed time) for the file. The file's last modification time will still be restored because if is contained in the zip file's central directory entry for each file.

xwrUnicodeData constant

Value

313

Description

Could not restore a file's Unicode filename from the zip file because the Unicode descriptor's data is corrupted.

Tips

The file will be unzipped with its regular ANSI filename.

xwrUnicodeSize constant

Value

314

Description

Could not restore a file's Unicode filename from the zip file because the Unicode descriptor's size is invalid.

Tips

The file will be unzipped using its regular ANSI filename.

xwrSecuritySize constant

Value

309

Description

Could not restore a file's security information from the zip file because the security descriptor's size is invalid.

Tips

The security header contains invalid or corrupted data. In this case the library will not restore a file's security information, but the file will be unzipped anyway.

ZipOpenedFiles property

Description

The ZipOpenedFiles property determines whether files that are currently opened for writing by another process or application should be skipped or not when running the Zip method.

Setting the ZipOpenedFiles property to False will cause any files currently opened for write operations to be skipped. The SkippingFile event is triggered whenever a file is skipped in this way.

Setting the ZipOpenedFiles property to True will allow Xceed Zip to attempt to zip the file despite it being opened for writing. When in this mode, you should verify that nothing is actually being written to the file at the moment it is being zipped, because the file may be zipped in an inconsistent state.

Data type

Boolean

Default value

False

Applicable methods

Zip

xcdUnzipDestination enumeration

```
xcdUnzipDestination
{
      xudDisk = 0,
      xudMemory = 1,
      xudMemoryStream = 2,
}
```

This enumeration is used by the UnzipPreprocessingFile property. Explanations on each parameter can be found in that property's help topic.

Help file revision

Xceed Zip Compression Library v4.0
Windows Help File Last Revised on April 15, 1999

xsrIncluded constant

Value

0

Description

The file is not being skipped.

Tips

This constant exists because the UnzipPreprocessingFile or ZipPreprocessingFile events can indicate that a file is excluded or not, and when they indicate that a file is excluded (i.e.: the bExcluded parameter is True) the xReason parameter contains the xsrIncluded constant (0) – the file is not excluded, and therefore not skipped. It's a filler constant.

xsrFilesToExclude constant

Value

1

Description

The filename matched an entry in the FilesToExclude property.

xsrSkipExisting constant

Value

2

Description

The file already exists in the destination, and the SkipIfExisting property is set to True.

xsrSkipNotExisting constant

Value

3

Description

The file does not already exist in the destination, and the SkipIfNotExisting property is set to True.

About Xceed Zip control skipping reasons

The following constants define the possible values of SkippingFile event's xReason parameter, as well as the xReason parameter provided by the ListingFile, PreviewingFile, UnzipPreprocessingFile and ZipPreprocessingFile events.

The declarations for these constants are built into the ActiveX control's interface and should automatically be accessible by your code.

Values for the xReason parameter that range between 1 and 99 are skipping reasons generated for files that were skipped due to the settings of the filtering properties or the SkipIf* properties. Values of 100 or more are skipping reasons generated because of an error while processing the file.

Possible values for the xReason parameter:

- xsrIncluded (0)
- xsrFilesToExclude (1)
- xsrSkipExisting (2)
- xsrSkipNotExisting (3)
- xsrSkipOlderDate (4)
- xsrSkipOlderVersion (5)
- xsrRequiredAttributes (6)
- xsrExcludedAttributes (7)
- xsrMinDate (8)
- xsrMaxDate (9)
- xsrMinSize (10)
- xsrMaxSize (11)
- xsrSkipUser (12)
- xsrDuplicateFilenames (13)
- xsrSkipReplace (14)
- xsrUpdateWithoutTemp (15)
- xsrInvalidDiskNumber (100)
- xsrFolderWithSize (101)
- xsrWriteFile (102)
- xsrOpenFile (103)
- xsrReadFile (104)
- xerMoveFile (105)
- xsrInvalidPassword (106)
- xsrInvalidCRC (107)
- xsrInvalidUncompSize (108)
- xsrCentralHeaderData (109)
- xsrLocalHeaderData (110)
- xsrDescriptorHeaderData (111)
- xsrCreateFolder (112)
- xsrAccessDenied (113)
- xsrCreateFile (114)
- xsrUpdateWithoutTemp (115)
- xsrRenamedToExisting (116)
- xsrVolumeWithSize (117)
- xsrCannotSetVolumeLabel (118)
- xsrCentralHeaderNotFound (119)
- xsrUnzipDiskFull (120)
- xsrCompress (121)
- xsrUncompress (122)

xsrSkipOlderDate constant

Value

4

Description

The file's date is older than the date of the file already existing in the destination, and the SkipIfOlderDate property is set to True.

xsrSkipOlderVersion constant

Value

5

Description

The file's version resource information is older than the version resource information of the file already existing in the destination, and the SkipIfOlderVersion property is set to True.

xsrRequiredAttributes constant

Value

6

Description

The file does not have all the required attributes listed in the RequiredFileAttributes property.

xsrExcludedAttributes constant

Value

7

Description

The file has one or more of the attributes listed in the ExcludedFileAttributes property.

xsrMinDate constant

Value

8

Description

The file's date is smaller than the date specified in the MinDate property.

xsrMaxDate constant

Value

9

Description

The file's date is greater than the date specified in the MaxDate property.

xsrMinSize constant

Value

10

Description

The file's size is smaller than the size specified in the MinSize property.

xsrMaxSize constant

Value

11

Description

The file's size is greater than the size specified in the MaxSize property.

xsrSkipUser constant

Value

12

Description

Your application set the bExcluded flag to True for this file during the occurrence of the file's ZipPreprocessingFile or UnzipPreprocessingFile event.

xsrDuplicateFilenames constant

Value

13

Description

The file has the same name and path as another file already selected to be zipped or to remain in the zip file.

Tips

This can result from including files on disk recursively while discarding path information (example, when the ProcessSubfolders property is set to True and you have specified wildcards and you have set the PreservePaths property to False).

xsrInvalidDiskNumber constant

Value

100

Description

The reported disk number for the file is invalid. This can happen if the zip file's central directory indicates that the file lies on a disk numbered above 32767.

Tips

The zip file is corrupted.

xsrFolderWithSize constant

Value

101

Description

The file is reported as being a folder with a size in bytes other than 0.

Tips

The zip file is corrupted. A file or folder with this type of problem is simply not unzipped.

xsrWriteFile constant

Value

102

Description

Error writing data to the file.

Tips

The library was able to open the file for writing, and was able to verify that there was enough disk space free. However, it still could not write data to the file. This can happen in exceptional situations, like when a user removes a floppy from the drive while the library is writing data to a file on it.

xsrOpenFile constant

Value

103

Description

Could not open the file for reading.

Tips

The file may currently be opened for reading in exclusive mode or for writing by another application. Setting the ZipOpenedFiles property to True may in some cases allow the library to open the file for reading even though it is currently opened for writing by another application.

xsrReadFile constant

Value

104

Description

Error reading data from the file.

Tips

The library was able to open the file for reading, but was not able to complete a read operation on the file. This is a severe error, and could be caused by circumstances such the removal of a floppy disk from the drive being read from, or an untimely disconnection of a network drive being read from.

xsrMoveFile constant

Value

105

Description

Cannot move a temporary file to the real file's destination.

Tips

When the library is instructed to compare version resource information before unzipping files (because the SkipIfOlderVersion property is set to True), it unzips each file into the temporary folder (see the TempFolder property for details) in order to compare its version resource information with the destination file's version resource information. The xsrMoveFile skipping reason can occur when the file is indeed going to replace the other file, but the library is unable to overwrite the older file with the new one just unzipped into the temporary folder. This may be a file security or permission problem.

xsrInvalidPassword constant

Value

106

Description

The file was encrypted and the library was unable to decrypt the file with the password provided.

Tips

The password in the EncryptedPassword property was invalid for this file, as well as any passwords provided to the InvalidPassword event if it was implemented.

xsrInvalidCRC constant

Value

107

Description

The calculated CRC (checksum) for the file does not match the CRC indicated in the zip file.

Tips

After the file was unzipped, the CRC calculated by the library does not match the CRC stored in the zip file. The purpose of the stored CRC is to allow the unzipping application or library to determine if the unzipped data has been restored to its original state. Therefore, getting this skipping reason means the f uncompressed data is corrupted. The file is unzipped anyway, so you can recover as much of it as you can.

xsrInvalidUncompSize constant

Value

108

Description

The file's real uncompressed size does not match the uncompressed size indicated in the zip file.

Tips

After the file was unzipped, the uncompressed size of the file differed from the uncompressed size stored in the zip file. Therefore, this is a good indication that the file was not recreated exactly the same as the original. Therefore, the file should be considered to be corrupted, to be incomplete or to contain extra data at the end.

xsrCentralHeaderData constant

Value

109

Description

The file's entry in the central directory contains conflicting information when compared to the file's local header.

Tips

This particular skipping reason can happen when you are using the unzip-without-last-disk unzipping mode and the zip file contains inconsistent data for this file.

xsrLocalHeaderData constant

Value

110

Description

The file's local header contains conflicting information when compared to the file's entry in the central directory.

Tips

The zip file contains inconsistent data for this file.

xsrDescriptorHeaderData constant

Value

111

Description

The file's descriptor header contains conflicting information when compared to the file's entry in the central directory.

Tips

The zip file contains inconsistent data for this file.

xsrCreateFolder constant

Value

112

Description

Could not create the folder that this file is being unzipped to.

Tips

The folder that cannot be created may be a part of the path specified in the unzipToFolder property, or it may be a folder that is part of the path stored in the zip file for the file being skipped.

xsrAccessDenied constant

Value

113

Description

The file being unzipped could not be written because the already existing file it was about to replace could not be deleted by the library.

Tips

This may be a file security or permission problem.

xsrCreateFile constant

Value

114

Description

The file being unzipped could not be created for writing in the destination unzipping location.

Tips

This may be a file security or permission problem. This can also happen if the destination filesystem or folder contains too many files.

xsrUpdateWithoutTemp constant

Value

15

Description

Could not update or replace a file in the zip file without using a temp file.

Tips

This can happen only when you have set the UseTempFile property to False and are adding files that already exist in the target zip file. The library is able to append files to an already existing zip file, without using a temporary file – but if some files already exist in the zip file and need to be replaced, then the library will skip those files. You can remedy the situation by turning on the use of temporary files with the UseTempFile property.

xsrRenamedToExisting constant

Value

116

Description

A file was renamed by the application (through the use of the ZipPreprocessingFile event) to a filename and path that already exists in the zip file.

Tips

This can only happen when zipping files.

xsrVolumeWithSize constant

Value

117

Description

The volume label is reported having a size in bytes other than 0.

Tips

The zip file is corrupted. The volume label is not restored to the destination unzipping location's drive.

xsrCannotSetVolumeLabel constant

Value

118

Description

Cannot set the volume label on the destination drive.

Tips

This can happen when the zip file contains a volume label, and the library is unable to restore it to the destination unzipping location drive (determined by the UnzipToFolder property).

Contents

Click on the topic you want to jump to.

xerWarnings constant

Value

526

Description

There were warnings during the operation. The Warning event is triggered whenever a warning occurs during an operation. You can write a handler for the Warning event if you want to know what the problems were and which files they were associated with.

About the Self-Extractor Module properties

The following Xceed Zip control properties are used by the Zip and Convert methods to configure the self-extracting zip files they create.

Most of these properties contain default values that are appropriate for self-extracting zip files. However, you will most likely want to change a few things, such as the title and introduction message of the self-extracting zip files you create, so you should browse the following properties to be sure you are getting the most from the self-extracting zip files this library can create.

List of the Xceed Zip control's properties related to the Self-Extractor Module:

- SfxButtons
- SfxDefaultPassword
- SfxDefaultUnzipToFolder
- SfxExecuteAfter
- SfxExistingFileBehavior
- SfxExtensionsToAssociate
- SfxIconFilename
- SfxInstallMode
- SfxMessages
- SfxProgramGroup
- SfxProgramGroupItems
- SfxReadmeFile
- SfxStrings

About the Self-Extractor Module methods

The Xceed Zip control provides the following methods to accomplish the tasks of creating self-extracting zip files.

List of the Xceed Zip control's methods related to creating self-extracting zip files:

- SfxAddExtensionToAssocitate
- SfxAddProgramGroupItem
- SfxClearButtons
- SfxClearMessages
- SfxClearStrings
- SfxResetButtons
- SfxResetMessages
- SfxResetStrings

The following two methods are used to create the self-extracting zip file, or to convert a regular zip file into a self-extracting zip file. They are part of the Xceed Zip control:

- Zip
- Convert

About the Self-Extractor Module enumeration types

The following enumerations are built into the Xceed Zip control's ActiveX interface and are used by some of the properties related to the Self-Extractor Module configuration properties.

List of the Xceed Zip control's enumeration types related to the Self-Extractor Module:

- xcdSfxButtons
- xcdSfxExistingFileBehavior
- xcdSfxStrings
- xcdSfxMessages

Converting a zip file into a self-extracting zip file

Adding self-extracting zip file capability to an already existing zip file

To add self-extracting capability to an already created zip file, perform the following 4 steps:

- **Specify the zip file to convert**. To do this, set the ZipFilename property.
- **Specify the self-extractor binary to use.** To do this, set the SfxBinaryModule property.
- **Tell Xceed Zip to start converting**. To do this, call the Convert method.
- **Make sure it worked successfully**. To do this, check the Convert method's return value.

Here is sample code for these 4 steps:

```
' Don't forget to put an Xceed Zip control on a form
Dim ResultCode As xcdError
XceedZip1.ZipFilename = "c:\zip files\sample.zip"
XceedZip1.SfxBinaryModule = "xcdsfx32.bin"
ResultCode = XceedZip1.Convert("c:\zip files\sample.exe")
If ResultCode = xerSuccess Then
    MsgBox "Converted successfully."
Else
    MsgBox XceedZip1.GetErrorDescription(xvtError, ResultCode)
EndIf
```

Things you should consider

You will probably want to customize the self-extracing zip files you create. For example, you may want to change the self-extracting zip file's title and introduction message, or have it suggest a default unzipping folder to the user running it. There is a group of properties that start with "Sfx" designed to let you do just that. See the About the Self-Extractor Module properties topic to find out what you can customize.

When you are converting an existing zip file into a self-extracting zip file, you can't modify the contents of the existing zip file. Consult the Creating a new self-extracting zip file topic if you want to know what kind of control you're missing when you aren't creating the self-extracting zip file directly from the source files to zip up.

Compress method

Description

The Compress method compresses data entirely in memory, without dealing with files. It supports streaming data and can also encrypt data while it is compressing it.

Declaration

```
Method Compress(vaSource As Variant, vaCompressed As Variant, bEndOfData As Boolean)
As xcdCompressionError
```

Parameters

Parameter	Description
vaSource	The data to compress. The variant can contain a string or a byte array.
vaCompressed	The compressed data is placed here, in the form of a byte array variant. The content of the vaCompressed parameter is always cleared before writing the compressed data to it.
bEndOfData	Set this parameter to True if you only have a single block of data to compress and require that the compressed data be immediately written to the vaCompressed parameter. Set this parameter to False if you are dealing with streaming data. The bEndOfData parameter allows you to provide the compression engine with data to compress as it becomes available. Whenever you have some new data to compress, put your data in the vaSource parameter, set bEndOfData to False, then call the Compress method. When the call completes, the compression engine may or may not have writen any compressed data to the vaCompressed parameter. If there is data in the vaCompressed parameter, you must grab it before calling the Compress method again, because if there is new compressed data, it will overwrite data currently present in the vaCompressed parameter. When your last block of data is sent to the Compress method in this fashion, set bEndOfData to True. The compression engine will flush all remaining compressed data to the vaCompressed parameter's variable.

Return values

See the xcdCompressionError topic for a description of the possible return values.

Remarks

If you want to encrypt the data while it is being compressed, set the EncryptionPassword property.

To control the amount of compression applied, set the CompressionLevel property.

The compressed data contains a 32-bit checksum that is used by the Uncompress method to make sure that the data uncompresses properly and is identical to the original uncompressed data sent to the Compress method.

If you are compressing a stream of data, and you want to force the compression engine to flush its output buffers right away without obtaining more uncompressed data, you can do this by calling the Compress method with an empty vaSource variable.

Applicable properties

The Xceed Compression control's EncryptionPassword, CompressionLevel properties.

Events triggered

None

Uncompress method

Description

The Uncompress method decompresses data entirely in memory. The Uncompress method deals with data that was compressed by the Compress method, and does not deal with zip files or zipped data. It works the same was as the Compress method does, and supports the same streaming data and encryption capabilities.

Declaration

```
Method Uncompress(vaSource As Variant, vaUncompressed As Variant, bEndOfData) As
xcdCompressionError
```

Parameters

Parameter	Description
vaSource	The data to uncompress. The variant can contain a string or a byte array.
vaUncompressed	The uncompressed data is placed here, in the form of a byte array variant. The content of the vaUncompressed parameter is always cleared before writing the uncompressed data to it.
bEndOfData	Set this parameter to True if you only have a single block of data to uncompress and require that the uncompressed data be immediately written to the vaUncompressed parameter. Set this parameter to False if you are dealing with streaming data. The bEndOfData parameter allows you to provide the decompression engine with data to uncompress as it becomes available. Whenever you have some new data to uncompress, put your data in the vaSource parameter, set bEndOfData to False, then call the Uncompress method. When the call completes, the decompression engine may or may not have writen any uncompressed data to the vaUncompressed parameter. If there is data in the vaUncompressed parameter, you must grab it before calling the Uncompress method again, because if there is new uncompressed data, it will overwrite data currently in the vaUncompressed parameter. When your last block of data is sent to the Uncompress method in this fashion, set bEndOfData to True. The decompression engine will flush all remaining uncompressed data to the vaUnompressed parameter's variable.

Return values

See the xcdCompressionError topic for a description of the possible return values.

Remarks

If the data was encrypted when it was compressed, you will have to provide the decryption password by setting the EncryptionPassword property. If you do not do this, you will get the xceInvalidPassword return value.

The compressed data contains a 32-bit checksum that is used by the Uncompress method to make sure that the data uncompresses properly and is identical to the original uncompressed data sent to the Compress method. If the data is uncompressed and has a bad checksum, you will get the xceChecksumFailed return value.

Applicable properties

The Xceed Compression control's EncryptionPassword property.

Events triggered

None

License method

Description

The License method must be called to license the control for runtime use when you are instantiating the control dynamically. This applies, for example, when using VB's CreateObject() call or the COM API call CoCreateInstance. You do not have to call this method when the control is placed on a form or if you are instantiating the control with the following VB code:

```
Dim WithEvents MyZipInstance As XceedZip
Set MyZipInstance = New Xceed Zip
```

The License method should be the first method you call after instantiating the Xceed Zip or Xceed Compression object.

If you do not call this method when dynamically instantiating the control with the CreateObject() or CoCreateInstance or similar methods of instantiating an object, all Xceed Zip methods will not work and will immediately return the xerNotLicensed error code.

When the control is being used on an HTML document with the <OBJECT> tag, a License Package File (LPK) should be used to provide the license key. You can obtain more information and an LPK creation kit from http://www.microsoft.com/workshop/components/activex/licensing.asp#LPKFILES or from http://www.microsoft.com/gallery/tools/lpktool/lpktool.asp.

Declaration

```
Method License(sLicense As String) As Boolean
```

Parameters

sLicense

Provide your license key here. Only registered users have a license key, which can be found in the LICENSE.TXT file located in the folder where the registered Xceed Zip Compression Library v4.0 package was installed. Users of the free trial version do not need to call the License method, because it is only required for applications that are being distributed, and the free trial version is intended to be used only on the development machine for evaluation purposes.

Return values

Returns True if the license key was accepted, False if the license key was invalid.

Remarks

The License method is required because instantiation methods such as CreateObject() and CoCreateInstance() do not cooperate with the licensing scheme provided by IclassFactory2.

License method

Description

The License method must be called to license the control for runtime use when you are instantiating the control dynamically. This applies, for example, when using VB's CreateObject() call or the COM API call CoCreateInstance. You do not have to call this method when the control is placed on a form or if you are instantiating the control with the following VB code:

```
Dim WithEvents MyZipInstance As XceedZip
Set MyZipInstance = New Xceed Zip
```

The License method should be the first method you call after instantiating the Xceed Zip or Xceed Compression object.

If you do not call this method when dynamically instantiating the control with the CreateObject() or CoCreateInstance or similar methods of instantiating an object, all Xceed Zip methods will not work and will immediately return the xerNotLicensed error code.

When the control is being used on an HTML document with the <OBJECT> tag, a License Package File (LPK) should be used to provide the license key. You can obtain more information and an LPK creation kit from http://www.microsoft.com/workshop/components/activex/licensing.asp#LPKFILES or from http://www.microsoft.com/gallery/tools/lpktool/lpktool.asp.

Declaration

```
Method License(sLicense As String) As Boolean
```

Parameters

sLicense

Provide your license key here. Only registered users have a license key, which can be found in the LICENSE.TXT file located in the folder where the registered Xceed Zip Compression Library v4.0 package was installed. Users of the free trial version do not need to call the License method, because it is only required for applications that are being distributed, and the free trial version is intended to be used only on the development machine for evaluation purposes.

Return values

Returns True if the license key was accepted, False if the license key was invalid.

Remarks

The License method is required because instantiation methods such as CreateObject() and CoCreateInstance() do not cooperate with the licensing scheme provided by IclassFactory2.

CompressionLevel property

Description

The CompressionLevel property controls the amount of compression to be applied to data being compressed by the Compress method. The greater the amount of compression applied, the greater the time it takes to perform the compression.

Data type

xcdCompressionLevel

Possible values

Value	Meaning
xclNone (0)	No compression. The Compress method will still work, but the data won't be smaller. The data will have a CRC checksum, though.
xclLow (1)	Minimum compression. This setting takes the least amount of time to compress data. When compared to the XcdMedium setting, this setting usually gives noticeably faster compression times in exchange for about 1 to 15% larger compressed data.
xclMedium (6)	Normal compression. This is the best balance between the time it takes to compress data and the compression ratio achieved.
xclHigh (9)	Maximum compression. This setting achieves the best compression ratios that the Zip deflate compression algorithm is capable of producing. When compared to the xcdMedium setting, this setting significantly increases compression time in exchange for about 1-3% smaller compressed data. We recommend that you use this setting only when you really need to achieve the smallest possible compressed data and when compression time is unimportant.

Default value

xclMedium

Browsable

Yes

Applicable methods

Compress

EncryptionPassword property

Description

The EncryptionPassword property contains the case-sensitive password to be used for encrypting or decrypting data with the Compress and Uncompress methods.

If the EncryptionPassword property is left empty before executing the Compress method, encryption will not be applied. If the data you are uncompressing was compressed with an encryption password, then the EncryptionPassword property must contain a valid decryption password for the data to uncompress, otherwise an error will be returned.

Passwords can be up to 80 characters long.

Data type

String

Default value

Empty

Browsable

Yes

Remarks

The encryption algorithm used by the Xceed Zip Compression Library is the same one used by the PKZip 2.04g program. This type of encryption can resist attacks by amateurs if the password is well chosen and long enough (at least 16 characters) but it will probably not resist attacks by determined users or experts. Therefore, we suggest that you always use long passwords, and when possible, do not rely solely on this encryption system to protect sensitive data.

Applicable methods

Compress, Uncompress

xcdCompressionError enumeration

```
xcdCompressionError
{
      xceSuccess = 0,
      xceSessionOpened = 1000,
      xceInitCompression = 1001,
      xceCompression = 1003,
      xceInvalidPassword = 1004,
      xceChecksumFailed = 1005,
      xceDataRemaining = 1006
      xceNotLicensed = 1007,
      xceBusy = 1008
}
```

This enumeration is used Compress and Uncompress methods. Explanations on each parameter can be found in the About the Xceed Compression control error codes help topic.

Streaming

Streaming data means data that is readable or obtainable only in a steady stream or in block portions of sequential data. For example, a file's data arriving through modem is considered streaming data because you only get access to the data in a sequential fashion, as it becomes available. A file already completely transferred is no longer streaming data – you can access the entire file at will.

xerDiskNotEmptyAbort constant

Value

528

Description

The library was instructed to abort the current operation during the occurrence of the DiskNotEmpty event.

Tips

This error occurs when the DiskNotEmpty event's xAction parameter is set to xnaAbort.

xwrInvalidFileCount constant

Value

319

Description

The zip file's central directory contains less entries than the amount indicated in the zip file's "end-of-central-directory" header.

Tips

The zip file is damaged.

Zipping files from memory directly to a zip file

Compression from memory to a zip file

Xceed Zip allows you to compress blocks of memory into files in a zip file. To accomplish this, perform the following 6 steps:

- **Specify the zip file to create or add files to**. To do this, set the ZipFilename property.
- **If you also have real files to zip up as well as blocks of memory, you must specify these real files.** To do this, set the FilesToProcess property.
- **Provide the filename(s) and file information for the block(s) of memory to compress into the zip file.** To do this, write a handler for the QueryMemoryFile event.
- **Tell Xceed Zip to start zipping**. To do this, call the Zip method.
- **Provide the data for the filename(s) you specified to compress from memory.** To do this, write a handler for the ZippingMemoryFile event.
- **Make sure it worked successfully**. To do this, check the Zip method's return value.

Here is sample code for these 6 steps (excluding the code for the QueryMemoryFile and ZippingMemoryFile events):

```
' Don't forget to put an Xceed Zip control on a form
Dim ResultCode As xcdError
XceedZip1.ZipFilename = "c:\outgoing\pictures.zip"
XceedZip1.FilesToProcess = "c:\graphics\cars\*.bmp"
ResultCode = XceedZip1.Zip
If ResultCode = xerSuccess Then
    MsgBox "Files zipped successfully."
Else
    MsgBox XceedZip1.GetErrorDescription(xvtError, ResultCode)
EndIf
```

Here is sample code to put into the QueryMemoryFile event handler. This sample assumes you only have a single file to zip up from memory:

```
sFilename = "Readme.txt" ' necessary
bFileProvided = (lUserTag <> 0) ' lUserTag is 0 the first time QueryMemoryFile
    is triggered, 1 the second time, 2 the third time, etc. So this line of
    code will only set bFileProvided to True once.
```

Here is sample code to put into the ZippingMemoryFile event handler:

```
vaDataToCompress = "Welcome to my special collection of " +
    "car pictures. These .bmp files can be loaded in MS Paint..."
vaEndOfData = True
```

Unzipping files from a zip file directly to memory

Decompressing from a zip file directly to memory

When unzipping files, Xceed Zip allows you selectively determine the way it will be unzipped. You can chose to unzip files to disk (the default), to unzip files into a single block of memory, or to provide you with streaming data (blocks of uncompressed data as it becomes available from the decompression engine). To unzip files to memory, perform the following 6 steps.

- **Specify the zip file to unzip files from**. To do this, set the ZipFilename property.
- **Specify the folder where to unzip files to (even if you are unzipping all files to memory)**. To do this, set UnzipToFolder property.
- **Tell Xceed Zip to start unzipping**. To do this, call the Unzip method.
- **Tell Xceed Zip which files you want it to unzip directly to memory.** To do this, write a handler for the UnzipPreprocessingFile event.
- **Get the uncompressed file data.** To do this, write a handler for the UnzippingMemoryFile event.
- **Make sure it worked successfully**. To do this, check the Unzip method's return value.

Here is sample code for these 6 steps (excluding the code for the UnzippingMemoryFile event):

```
' Don't forget to put an Xceed Zip control on a form
Dim ResultCode As xcdError
XceedZip1.ZipFilename = "c:\incoming\searchresults.zip"
XceedZip1.UnzipToFolder = "c:\other files\"
ResultCode = XceedZip1.Unzip
If ResultCode = xerSuccess Then
    MsgBox "Files unzipped successfully."
Else
    MsgBox XceedZip1.GetErrorDescription(xvtError, ResultCode)
EndIf
```

Here is sample code to put into the UnzipPreprocessingFile event handler. This sample assumes you only want two particular files to be unzipped to memory, and the rest to go to "c:\other files":

```
if sFilename = "results.txt" or sFilename = "query.txt" then
   xDestination = xudMemory
endif
```

Here is sample code to put into the UnzippingMemoryFile event handler. This sample assumes you have declared the "results" and "query" variables globally elsewhere than in the UnzippingMemoryFile event handler:

```
Dim Results As String
Dim Query As String
if sFilename = "results.txt" then Results = baData
if sFilename = "query.txt" then Query = baData
```

Event handler

Event-specific, user-provided code that is executed whenever the event is triggered. For example, if you want to play a sound whenever a button is clicked, you would write code to play the sound in the event handler for the button's click event.

xerFilesSkipped constant

Value

527

Description

Some of the files to process were skipped (not processed) during the operation because of errors.

Tips

The SkippingFile event is triggered whenever a file is skipped for any reason during an operation. Sometimes files are skipped because of errors (such as a read or write problem), and other times they are skipped because your application instructed Xceed Zip to skip them (using the bExcluded parameter in the ZipPreprocessingFile or UnzipPreprocessingFile event, for example). You will only get the xerFilesSkipped result code if one of more files were skipped because of errors.

You can write a handler for the SkippingFile event if you want to know what the errors were and which files they were associated with.

xerRemoveWithoutTemp constant

Value

529

Description

The RemoveFiles method was called, but the UseTempFile property was set to False.

Tips

The library must use a temporary file in order to remove files from a zip file. Set the UseTempFile property to True to remedy the situation by allowing the library to use a temp file.

Installation instructions for Visual Basic 4.0 (32-bit)

The Xceed Zip Compression Library setup utility will copy all the necessary files to your system and place the XCEEDZIP.DLL in the Windows System folder and "register" the control with the operating system. The only thing left to do is to make the control available to your VB 4.0 projects. To do this, perform the following steps:

- Start the 32-bit Visual Basic 4.0.
- Select Custom Controls... from the Tools menu.
- In the Show box, select the Controls checkbox.
- An entry named "Xceed Zip Compression Library v4.0" entry should appear in the "available controls" list.
- Make sure the checkbox next to the "Xceed Zip Compression Library v4.0" entry is selected. Click on the OK button and the Xceed Zip and Xceed Compression icons should now appear in your toolbox.

If the control is not in the "available controls" list, perform the following steps:

- Try reinstalling the Xceed Zip Compression Library from its setup program. Continue only if that didn't help.
- Select the Browse... button from the Custom Controls dialog.
- Find and select the XCEEDZIP.DLL file (use the one in the Windows System folder that was copied there by the setup program) then click on the OK button.
- The control should now appear in the components list.

Installation instructions for Visual C++ 4.0 (MFC projects)

The Xceed Zip Compression Library setup utility will copy all the necessary files to your system and place the XCEEDZIP.DLL in the Windows System folder and register it with the operating system. The only thing left to do is to create a wrapper class for your MFC projects (assuming you're working with MFC) and make the control available to your VC++ 4.0 projects. To do this, perform the following steps:

- Start Visual C++ 4.0 (developer studio)
- Open your MFC AppWizard application project, or create a new one by selecting the New... menu item from the File menu, and choosing the Project Workspace option. Select MFC AppWizard and follow the Wizards instructions to create the project. Be sure that you select the OLE Controls option for the question What OLE support would you like to include?
- Select the Component... menu item from the Insert menu.
- Select the OLE Controls tab in the Component Gallery dialog.
- The Xceed Zip control icon appear in the list of controls displayed in the OLE Controls tab in the Component Gallery dialog.
- Select the Xceed Zip control icon, then click the Insert button in the Component Gallery dialog to insert the control in your project. The Confirm Classes dialog should appear.
- Click OK to close the Confirm Classes dialog.
- Click the Close button to close the Component Gallery dialog.
- The XCEEDZIP.CPP and XCEEDZIP.H files should now be part of your project. You will now be able to instantiate the Xceed Zip object in your project, and see the Xceed Zip icon available in your Controls toolbar when designing dialog boxes.

If the Xceed Zip control icon does not appear in the list of controls displayed in the OLE Controls tab in the Component Gallery dialog:

- Try reinstalling the Xceed Zip Compression Library from its setup program. Continue only if that didn't help.Select the Customize... button.
- Select the OLE Controls category in the Customize Component Gallery dialog.
- Select the Import... button in the Customize Component Gallery dialog.
- Find and select the XCEEDZIP.DLL file (preferably the one in the Windows System folder that was copied there by the Xceed Zip setup program), then click on the OK button.
- Click the OK button in the Customize Component Gallery dialog.
- The Xceed Zip control icon should now appear in the list of controls.

Installation instructions for Visual C++ 5.0 (MFC projects)

The Xceed Zip Compression Library setup utility will copy all the necessary files to your system and place the XCEEDZIP.DLL in the Windows System folder and register it with the operating system. The only thing left to do is to create a wrapper class for your MFC projects (assuming you're working with MFC) and make the control available to your VC++ 5.0 projects. To do this, perform the following steps:

- Start Microsoft Visual C++ 5.0
- Open your MFC AppWizard application project, or create a new one by selecting the New... menu item from the File menu, then selecting the Projects tab and choosing the MFC AppWizard item. Follow the Wizards instructions to create the project. Be sure that you select the ActiveX Controls option for the question "What other support would you like to include?"
- Select the Project menu
- Select the Add to project submenu, then select Components and Controls...
- In the Components and Controls Gallery dialog, look in the Registered ActiveX Controls folder for the "Xceed Zip Control 4.0" or the "Xceed Compression control" entry, then select it and click on the Insert button.
- Select the OK in response to the Insert this component? message box. The Confirm Classes dialog should appear.
- Click OK to close the Confirm Classes dialog.
- Click the Close button to close the Components and Controls Gallery dialog.
- The XCEEDZIP.CPP and XCEEDZIP.H files should now be part of your project. You will now be able to instantiate the Xceed Zip object in your project, and see the Xceed Zip icon available in your Controls toolbar when designing dialog boxes.

xerNotLicensed constant

Value

530

Description

This error occurs when the Xceed Zip control is dynamically instantiated without calling License method.

Tips

You must call the license method in certain cases of dynamic instantiation. Please see the documentation's License method topic to find out how to proceed.

xsrCentralHeaderNotFound constant

Value

119

Description

A file that was unzipped in unzip-without-last-disk mode did not to have an entry in the central directory.

Tips

The zip file is corrupted.

xsrCompress constant

Value

121

Description

An error occurred while the compression engine was compressing the file's data.

Tips

The file's data already written by the library is erased from the zip file.

xsrUncompress constant

Value

122

Description

An error occurred while the compression engine was uncompressing the file's data.

Tips

The file's compressed data is probably corrupted.

xsrUnzipDiskFull constant

Value

120

Description

The file could not be unzipped because the destination unzipping location's disk is full.

Tips

If the filesystem is not full, then you should check to make sure everything is working correctly with the drive, and check if there are any security or permission problems for accessing this drive.

Creating a spanned zip file

One extra thing to do

Except for the basic things you have to do to zip or unzip a file, the only extra thing you have to do to create or be able to unzip a spanned zip file is to write a handler for the InsertDisk event. The InsertDisk event is triggered by the library when a new disk must be inserted into the removable media drive in order to continue reading or writing a zip file. The Xceed Zip Compression Library never displays anything on the screen itself (a library should never take that initiative) so you have to display the message box yourself. The advantage is that you get to display whatever you want, any way you want it.

Here is sample code you can put in the InsertDisk event's handler for displaying a simple message box to insert a new or specific disk:

```
If nDiskNumber = xcdLastDisk then
    MessageBox "Please insert the last disk of the zip file set."
Else
    MessageBox "Please insert disk " + Str(nDiskNumber)
        + " of the zip file set."
EndIf
```

When this is done, the library will automatically span disks when you are writing to floppy disks and they get full, or will automatically request the appropriate disks when you are unzipping from a zip file that spans multiple disks.

The Xceed Zip Compression Library can also read and write zip files that are split onto multiple files in the same location (as opposed to a zip file split onto multiple disks). See the SplitSize property for information on how to do this.

Adding a progress bar

One extra thing to do

Except for the basic things you have to do to zip or unzip a file, the only extra thing you have to do to add a progress bar for the zip or unzip (or other) operation is to write a handler for GlobalStatus event which sets the "percentage" property of your progress bar. The GlobalStatus event provides you with status information on the entire zipping or unzipping operation, and one of the pieces of information it provides is the completion status, ranging from 0 to 100%. If you want to know the status of the operation on a file-by-file basis, use the FileStatus event instead.

Here is sample code you can put in the GlobalStatus event's handler:

```
MyProgressbar.value = nBytesPercent
```

The progress bar will then go from 0 to 100% as the zipping or unzipping is performed. You can reset it back to 0 after the library returns a result code from the zip or unzip method. The code is easy:

```
MyProgressbar.value = 0
```

That's it. Of course, you can do a lot more, so we do suggest you read the GlobalStatus property's help topic.

Macros for the self-extractor module properties

Some properties that used by the Self-Extractor Module allow macros to be used when specifying paths and filenames. The macros available are listed in each property's description.

A macro is a substring of two characters that has the format %x. The x is a letter representing one of the available macros (see below). The two characters representing the macro are replaced by a path only when the actual self-extracting Zip file is executed. Thus, the value of a macro usually depends on the particular settings of the Windows operating system the self-extracting Zip file is being run on.

All macros except for the %r macro include the drive as part of the path. The %v macro only contains the drive letter followed by the colon (:) character. None of the macros contain a trailing backslash.

Macros used by SfxStrings and SfxMessages properties are special macros that apply only to those properties. Only the macros common to multiple properties are listed below.

Here is a list of the common macros:

Macro	Description
%w	The location of the Windows folder.
%s	The location of the Windows System folder.
%t	The systems temporary folder.
%r	Random folder name that doesn't already exist. (the format of the random name is: _SFXxxx, where xxx is a random number)
%d	The unzipping folder, or the user-selected folder where files were unzipped to.
%e	The folder where the self-extractor is being run from.
%p	The Program Files folder (only available in Windows 95 and NT 4.x and up).
%v	The main system hard disk.

All macros except for the %r macro include the drive letter. The %v macro only contains the drive letter followed by the colon (:) character. None of the macros contain a trailing backslash.

xcdCompressionLevel enumeration

```
xcdCompressionLevel
{
      xclNone = 0,
      xclLow = 1,
      xclMedium = 6,
      xclHigh = 9
}
```

This enumeration is used by the CompressionLevel property. Explanations on each parameter can be found in that property's help topic.

xerProcessStarted constant

Value

1

Description

The method is executing its processing in the background.

Tips

When the BackgroundProcessing property is set to True, all Xceed Zip methods will return the xerProcessStarted constant to indicate that they are performing their work in the background. The real return value of the method will be provided by the ProcessCompleted event.

xsrSkipReplace constant

Value

14

Description

The file was skipped because the bReplaceFile parameter of the ReplacingFile event was set to False in the event handler.

Tips

Verify the code you have placed in the ReplacingFile event handler.

ReplacingFile event

Description

The ReplacingFile event is triggered whenever a file is about to be overwritten in the unzipping destination, or in the destination zip file.

The ReplacingFile event's bReplaceFile parameter allows you to determine whether or not to replace the file. The default value of the bReplaceFile parameter is False, so if you do not write a handler for this event, files will automatically be replaced. Setting the bReplaceFile parameter to True will cause the file to be overwritten.

Only the bReplaceFile parameter of the SkippingFile event is modifiable, all other parameters are there only to provide information about the file about the new file and the destination file.

If you want to rename files that already exist in the destination, then you should use the ZipPreprocessingFile or UnzipPreprocessingFile events. These events have a bExisting parameter which tells you whether or not the destination file exists, and allows you to perform many possible actions, including renaming the file or changing its path.

Declaration

Parameters in bold are modifiable

```
Event SkippingFile(sFilename As String, sComment As String, lSize As Long,
xAttributes As xcdFileAttributes, dtLastModified As Date, dtLastAccessed As Date,
dtCreated As Date, lDestSize As Long, xDestAttributes As xcdFileAttributes,
dtDestLastModified As Date, dtDestLastAccessed As Date, dtDestCreated As Date,
bReplaceFile as Boolean)
```

Parameters

Parameter	Description
sFilename	The path and filename of the destination file that may be replaced by the new file that has the same path and filename.
sComment	The new file's comment.
lSize	The new file's uncompressed size in bytes.
xAttributes	The new file's attributes.
dtLastModified	The new file's last modification date and time stamp.
dtLastAccessed	The new file's last accessed date and time stamp.
dtCreated	The new file's creation date and time stamp.
lDestSize	The destination file's uncompressed size in bytes.
iDestAttributes	The destination file's attributes.
dtDestLastModified	The destination file's last modification date and time stamp.
dtDestLastAccessed	The destination file's last accessed date and time stamp.
dtDestCreated	The destination file's creation date and time stamp.
bReplaceFile	Set this to True (default) to replace the destination file by the new one, or set it to False to keep the destination file.

ZipContentsStatus event

Description

The ZipContentsStatus event provides a status report when a zip file's contents are being scanned.

When a call is made to the Zip method to update or add files to an existing zip file, or when a call is made to the Convert method, the library must scan the existing zip file's central directory. This operation, although not processor or drive intensive, can still be lengthy when the zip file contains five to ten thousand files or more – especially if the zip file is located on a floppy disk or other slow drive or network connection. Therefore, the ZipContentsStatus event is useful in order to provide a progress or "working" indicator for this phase of the zipping and unzipping.

If you aren't adding to, updating or converting an already existing (very large) zip file, then providing a progress report with the GlobalStatus event is definitely sufficient.

None of the parameters are modifiable.

Declaration

```
Event ZipComment(lFilesTotal As Long, lFilesRead As Long, nFilesPercent As Integer)
```

Parameters

Parameter (Modifiable)	Description
lFilesTotal	Indicates the total number of files in the zip file that need to be scanned.
lFilesRead	The current number of files
nFilesPercent	The completion percentage (0% to 100%) based on how many files have been scanned so far.

Remarks

For zip files with 100 or more files in them, this event is triggered up to a maximum of 101 times (this reduces this event's overhead to negligible levels). In fact, it is triggered once for each possible value of the nFilesPercent parameter (from 0% to 100%). If the zip file has less than 100 files, it will be triggered once for each file in the zip file.

Applicable methods

Zip, ListZipContents

BackgroundProcessing property

Example

Description

The BackgroundProcessing property controls whether or not Xceed Zip methods are executed immediately or are run as background processes.

Normally this property is set to False, so method calls are not executed as background processes, and the entire operation is processed immediately and returns the real result code for the operation. When the BackgroundProcessing property is set to True, the operation starts immediately, however the ResultCode is also immediately set to xerProcessStarted and the code following the method call is executed. When the zip operation finally completes, the ProcessCompleted event is triggered, and provides the real result code for the operation, as well as statistics on the operation.

Data type

Boolean

Default value

False

Remarks

A separate thread is started by the library in order to accomplish the capability provided by setting this property to True. This property is useful when you need methods to be executed asynchronously.

Applicable methods

All methods except for the AddFilesToProcess, AddFilesToExclude, AddExtentionToAssociate and AddProgramGroupItem methods, which don't launch an operation at all.

BackgroundProcessing property example

As an example, consider that if the BackgroundProcessing property is left at its default value of False, and you run the Zip method, with code similar to this:

```
XceedZip1.FilesToProcess = "c:\my files\*"
...set other Xceed Zip properties
ResultCode = XceedZip1.Zip
...the rest of your code follows here
```

…the complete zip operation will be completed, then the ResultCode variable will be set to a value indicating whether or not the operation was succesful, and then the rest of your code will be executed.

However, if the BackgroundProcessing property is set to True, running the same code above will cause the zip operation to start, will immediately set the ResultCode variable equal to xerProcessStarted, will execute the rest of the code after the "ResultCode = XceedZip1.Zip" line, and later, when the zipping operation has completed, will trigger the ProcessCompleted event to let you know that it is finished, and to provide you with the result code (via the xResult parameter).

xceSuccess constant

Value

0

Description

This value indicates that the method's process executed 100% successfully.

xceSessionOpened constant

Value

1000

Description

The Compress (or Uncompress) methods were called one or more times with the bEndOfData parameter set to False, but were not called with the bEndOfData parameter set to True before a call to the Uncompress (or Compress) method was placed.

Tips

You must terminate a Compress or Uncompress session with the bEndOfData parameter set to True.

xceInitCompression constant

Value

1001

Description

The compression engine failed to initialize.

Tips

This is a critical error, unexplained error. Contact Xceed Software for further instructions if you get this error code.

xceCompression constant

Value

1003

Description

The compression or decompression engine returned an error.

Tips

If you are uncompressing data, the data may be corrupted, or it has been altered in some way from the time that it was obtained by the Compress method. Verify that the compressed data you are getting from the Compress method is exactly the same as the compressed data you are passing to the Uncompress method (bit for bit).

If you are compressing data, this should not happen (but is a possibility). Contact Xceed Software if it does. You may be passing an invalid type data type to the Compress method.

xceInvalidPassword constant

Value

1004

Description

The password provided in the EncryptionPassword property is invalid for the data passed to the Uncompress method.

xceChecksumFailed constant

Value

1005

Description

The 32-bit checksum calculated for the newly uncompressed data does not match the checksum value stored with the data.

Tips

The compressed data is either corrupted, or it has been altered in some way from the time that it was obtained by the Compress method. Verify that the compressed data you are getting from the Compress method is exactly the same as the compressed data you are passing to the Uncompress method (bit for bit).

xceDataRemaining constant

Value

1006

Description

The compressed data passed to the Uncompress method was succesfully uncompressed by the compression engine and passed the CRC test, however, not all the compressed data was used in order to produce the uncompressed output. In other words, the decompression engine expected less data than the amount passed to it.

Tips

Either too much data is being passed to the Uncompress method (you are providing extra, unnecessary data after the compressed data) or the data may possibly be corrupted (although this latter possibility is unlikely due to the checksum verification step).

CalculateCrc method

Description

The CalculateCrc method is provided as a convenience feature for developers that need to calaulate a CRC on data without compressing it at all. The checksum provided by the CalculateCrc method is the industry standard CRC-32 checksum used by the Zip format and the Zmodem communications protocol, for example.

This method allows you to calculate the CRC for an entire block of data in one call, or you can call it multiple times.

Declaration

```
Method CalculateCrc(vaData As Variant, lPreviousCrc As Long) As Long.
```

Parameters

Parameter	Description
vaData	The data to calculate the CRC-32 for. This is a variant of type string or byte array.
lPreviousCrc	When you call the CalculateCrc method and require the CRC-32 for an entire block of data, set this parameter to 0. This parameter is mainly used when you are making multiple calls to the CalculateCRC method for a very large block of data, or for streaming data. The return value of the previous CalculateCrc method call should be provided in this parameter so that the CRC-32 can continue to be updated for the rest of the data.

Return values

The return value contains the CRC-32 calculated for the data.

Applicable properties

None.

About the sample applications

If the getting started quickly section has topics that tell you what you have to do to accomplish basic compression-related tasks. However, real applications often have tougher problems to solve.

Xceed has provided you with various flavors of sample applications, to help you out.

The first type of sample application is named the Getting Started Sample Application. This sample application with source code is provided for various development environments, and provides you with a visual application that shows you, method-by-method, property-by-property, how to use the most important features of the library.

- Go to the Getting Started Sample Application for VB5 and VB6 topic

The second type of sample application is named the Zip Manager Sample Application. This is a complete Zip file manipulation application that can also create self-extracting zip files. It's kind of like WinZip™, but Xceed doesn't sell it, and you get the fully commented source code.

- Go to the Zip Manager Sample Application for VB5 and VB6 topic

Also included are sample applications to show C++ users how to use the controls to sink events in ATL and compress in memory compression.

- Go to the Visual C++ samples topic

About the getting started quickly section

This section of the manual has but one goal: To show you, as fast as possible, how to start using the library's major features.

If you don't see what you need here, the next step is to read the overview of the Xceed Zip control. From there you will probably jump to the methods or properties topic. If all else fails, use the "search" feature of the manual, or browse the sample applications. Feel free to contact Xceed Software at any time if you have questions.

Getting started quickly topics

Unzip method example for VB

The following code uses the Unzip method to unzip the file "README.TXT" from the zip FILE "C:\TEST\MY TEST.ZIP" into the directory "C:\TEST\UNZIPPED FILES". If the directory does not already exist, it will be created automatically. You should check the Unzip method's return value once you are done – if it is xerSuccess, the operation was succesful. Otherwise, look up the error code in the error codes list to find out what went wrong. This code assumes you have a button and an Xceed Zip control on a form, named Command1 and XceedZip1 respectively.

```
Sub Command1_Click()
    Dim ResultCode As xcdError
    ' All properties keep their default values except the four below
    XceedZip1.FilesToProcess = "readme.txt" ' The file to unzip
    XceedZip1.PreservePaths = False ' In case file is stored in the
        ' zip file with a path, we need to make sure the path is
        ' removed so that the file will match with "readme.txt"
    XceedZip1.UnzipToFolder = "c:\test\unzipped files"
    XceedZip1.ZipFilename = "c:\test\my test.zip"
    ' Start unzipping
    ResultCode = XceedZip1.Unzip
    ' Check the return value.
    If ResultCode <> xerSuccess Then
        MsgBox "Unsuccessful. Error # " + Str(nErr) + " occurred. " +
            "Description: " + XceedZip1.GetErrorDescription(xvtError,
    ResultCode)
    Else
        MsgBox "File(s) successfully unzipped."
    End If
End Sub
```

Unzip method example for Delphi

The following code uses the Unzip method to unzip the file "README.TXT" from the zip file "C:\TEST\MY TEST.ZIP" into the directory "C:\TEST\UNZIPPED FILES". If the directory does not already exist, it will be created automatically. You should check the Unzip method's return value once you are done – if it is xerSuccess, the operation was succesful. Otherwise, look up the error code in the error codes list to find out what went wrong. This code assumes you have a button and an Xceed Zip control on a form, named Button1 and XceedZip1 respectively.

```
procedure TForm1.Button1Click(Sender: TObject);
var
    ResultCode: xcdError;
begin
    { All properties keep their default values except the four below }
    XceedZip1.FilesToProcess := 'readme.txt'; { The file to unzip }
    XceedZip1.PreservePaths := False; { In case file is stored in the zip file
    with a path, we need to make sure the path is removed so that the file
    will match with 'readme.txt' }
    XceedZip1.UnzipToFolder := 'c:\test\unzipped files';
    XceedZip1.ZipFilename := 'c:\test\my test.zip';
    { Start unzipping }
    ResultCode := XceedZip1.Unzip;
    { Check the return value. If ResultCode = xerSuccess, it worked }
end;
```

RemoveFiles method example for VB

The following code deletes all files ending with ".TXT" from the zip file named "C:\TEST\MY TEST.ZIP". The example assumes you have a button and an Xceed Zip control on a form named Command1 and XceedZip1 respectively.

```
Sub Command1_Click()
    Dim ResultCode As xcdError
    ' All properties keep their default values except the two below
    XceedZip1.FilesToProcess = "*.txt"
    XceedZip1.ZipFilename = "c:\test\my test.zip"
    ' Start the delete operation
    ResultCode = XceedZip1.RemoveFiles
    ' Check the return value.
    If ResultCode <> xerSuccess Then
        MsgBox "Unsuccessful. Error # " + Str(nErr) + " occurred."
    Else
        MsgBox "File(s) succesfully deleted."
    End If
End Sub
```

RemoveFiles method example for Delphi

The following code deletes all files ending with ".TXT" from the zip file named "C:\TEST\MY TEST.ZIP". The example assumes you have a button and an Xceed Zip control on a form named Button1 and XceedZip1 respectively.

```
procedure TForm1.Button1Click(Sender: TObject);
var
    ResultCode: xcdError;
begin
    { All properties keep their default values except the two below }
    XceedZip1.FilesToProcess := '*.txt';
    XceedZip1.ZipFilename := 'c:\test\my test.zip';
    { Start the delete operation }
    ResultCode := XceedZip1.RemoveFiles;
    { Check the return value. If ResultCode = xerSuccess, it worked }
end;
```

ListZipContents method and ListingFile event example for VB

The first function of the following code lists all the files in the zip file "C:\TEST\MY TEST.ZIP" that finish with ".TXT". For each file listed, the ListingFile event will be triggered. The second function below is a handler written for the ListingFile event that displays each file listed, along with some other of the file's information, in a listbox. The code assumes you have placed a button, a listbox and an Xceed Zip control on a form, and named them Command1, List1 and XceedZip1 respectively.

We start with the handler for the button's click event. The code will start the listing.

```
Sub Command1_Click()
    Dim ResultCode As xcdError
    ' All properties keep their default values except the two below
    XceedZip1.FilesToProcess = "*.txt" ' process only .txt files
    XceedZip1.ZipFilename = "c:\test\my test.zip"
    ' Start listing
    ResultCode = XceedZip1.ListZipContents
    ' Check the result code to make sure it worked properly
    If (ResultCode <> xerSuccess) Then
        List1.AddItem "Errors/warnings received, or a file was skipped."
    End If
End Sub
```

And here's the handler for the ListingFile event. It will add each file's name and info to the listbox.

```
Private Sub XceedZip1_ListingFile(ByVal sFilename As String, ByVal sComment As
    String, ByVal lSize As Long, ByVal lCompressedSize As Long, ByVal
    nCompressionRatio As Integer, ByVal xAttributes As
    XceedZipLibCtl.xcdFileAttributes, ByVal lCRC As Long, ByVal dtLastModified
    As Date, ByVal dtLastAccessed As Date, ByVal dtCreated As Date, ByVal
    xMethod As XceedZipLibCtl.xcdCompressionMethod, ByVal bEncrypted As
    Boolean, ByVal lDiskNumber As Long, ByVal bExcluded As Boolean, ByVal
    xReason As XceedZipLibCtl.xcdSkippingReason)
    List1.AddItem "Filename: " + sFilename
    List1.AddItem "    Uncompressed size: " + Str(lSize)
    List1.AddItem "    Compressed size: " + Str(lCompressedSize)
    List1.AddItem "    Compression ratio: " + Str(nCompressionRatio)
End Sub
```

ListZipContents method and ListingFile event example for Delphi

The first function of the following code lists all the files in the zip file "C:\TEST\MY TEST.ZIP" that finish with ".TXT". For each file listed, the ListingFile event will be triggered. The second function below is a handler written for the ListingFile event that displays each file listed, along with some other of the file's information, in a listbox. The code assumes you have placed a button, a listbox and an Xceed Zip control on a form, and named them Command1, ListBox1 and XceedZip1 respectively.

We start with the handler for the button's click event. The code will start the listing.

```
procedure TForm1.Button1Click(Sender: TObject);
var
    ResultCode: xcdError;
begin
    { All properties keep their default values except the two below }
    XceedZip1.FilesToProcess := '*.txt';
    XceedZip1.ZipFilename := 'c:\test\my test.zip';
    { Start listing }
    ResultCode := XceedZip1.ListZipContents;
    { Check the result code to make sure it worked properly }
    if (ResultCode <> xerSuccess) then
    begin
        Listbox1.Items.Add('An error or warning occured, or files were
    skipped.');
    end;
 end;
```

And here's the handler for the ListingFile event. It will add each file's name and info to the listbox.

```
procedure TForm1.XceedZip1ListingFile(XceedZip: TXceedZip;
    const FileStats: TXcdFileStats);
begin
    ListBox1.Items.Add('Filename: ' + sFilename);
    ListBox1.Items.Add('    Uncompressed size: ' +
    IntToStr(lSize));
    ListBox1.Items.Add('    Compressed size: ' +
    IntToStr(lCompressedSize));
    ListBox1.Items.Add('    Compresion ratio: ' +
    IntToStr(nCompressionRatio));
end;
```

Installation instructions index

The Xceed Zip Compression Library setup utility will copy all the necessary files into the installation directory you selected and registered the XCEEDZIP.DLL ActiveX control in your Windows System folder. There are some additional steps you must take to complete the installation, based on which development language you are using.

Click on one of the following links for instructions specific to your language.

xceNotLicensed constant

Value

1007

Description

This error occurs when the Xceed Compression control is dynamically instantiated without calling the License method.

Tips

You must call the license method in certain cases of dynamic instantiation. Please see the documentation's License method topic to find out how to proceed.

xceBusy constant

Value

1008

Description

The current instance of the library is already running another operation.

Tips

Although the library is fully multi-threaded, each instance of the Xceed Compression control can only do one operation at a time. This makes event handling much clearer. The developer can create another instance of the library if it is necessary to do another, independent zipping or unzipping operation simultaneously.

Samples for Visual C++ users

Memory compression sample

The sample is located in "C:\PROGRAM FILES\XCEED ZIP\SAMPLES\OTHER\VC6\MEMORY COMPRESSION".

To use the "Memory Compression" sample, you will need Microsoft Visual C++ 6.0. The sample demonstrates how to:

- Import Xceed Zip's type library in a C++ project
- How to implement an event sink using ATL
- Perform memory compression with the Xceed Zip control

To use the sample, proceed as follows:

- Open the "MEMORY COMPRESSION.DSW" file in the MSVC IDE.
- The files XCEEDZIP.DLL and ZIPDISPIDS.H must be made available in the include path in order to compile the project. Current project settings include "\XCEEDZIP" in the include path, so you can either copy the two required files to this folder, or modify the include path accordingly.
- All the important calls are implemented in "MEMORY COMPRESSION.CPP".

Importing the type library

- The first thing that the sample demonstrates is how to import XceedZip's type library in a C++ project. The line "#import "XCEEDZIP.DLL" generates XCEEDZIP.TLH, which is the header file containing the declarations for the enumerations, classes, and other things contained in the type library. It also generates XCEEDZIP.TLI, which is the implementation of the methods declared in XCEEDZIP.TLH. Complete documentation on the #import directive can be found in the MSVC documentation.
- Looking at the main function shows how the wrapper classes generated by the #import directive can be used.
- Up to this point, we only used MSVC features (#import is a Microsoft C++ extension), with no MFC or ATL specific features.

Event sinking

- The second thing that the sample demonstrate is how to implement an event sink using ATL.
- We first need to include ATL's main header files. Look at stdafx.h, between the BEGIN/END ATL REQUIRED comments.
- Then, we need to declare a class that will derive either from IDispEventImpl<> or IDispEventSimpleImpl<>. The major difference between these two class templates is that the former uses information from the type library to determine the event handlers' information (return and parameter types), while the latter gets this information from user-provided structures.
- With Xceed Zip, we use IdispEventSimpleImple<> because we use enumeration types in many of the events, and IdispEventImpl<> does not handle that correctly.
- The first thing to do to implement an event is to add a member function to the sink class. This function must use _stdcall calling convention, and must be declared exactly as the event is defined in the event interface. **Tip: you can copy/paste the declaration from XCEEDZIP.TLH.**
- Next, a _ATL_FUNC_INFO structure must be declared for each event handler. This structure specifies the return and parameter types for the handler.
- Finally, a SINK_MAP must be declared in the sink class, using ATL macros. Each event handler in the sink class must be provided through a SINK_ENTRY_INFO() entry in the sink map.
- To use the sink object, all that needs to be done is to call DispEventAdvise() with a pointer to the object from which the events need to be handled.

xerInvalidSfxProperty constant

Value

531

Description

The SfxProgramGroupItems or the SfxExtensionsToRegister property contains an invalid item.

Tips

Verify that the string is properly formatted and that each item is separated by a vertical bar "|", a linefeed, or the return+linefeed characters.

Getting Started Sample App for Delphi 3 and 4

The Getting Started Sample Application for Delphi 3.0 and Delphi 4.0 can be found in the C:\PROGRAM FILES\XCEED ZIP\SAMPLES\GETTING STARTED\DELPHI3 folder (or the DELPHI4 folder).

To use the sample, load the STARTED.DPR project file, then run the program. The application itself explains everything to you on screen and with the help of pop-up tooltip boxes when you move the mouse over the controls in run mode.

Running multiple operations at the same time

The Xceed Zip Compression Library v4.0 ActiveX control supports both the Single-Threaded Apartment model (STA) and the Multi-Threaded Apartment model (MTA). The entire library's code is threadsafe, can be instantiated multiple times in order to perform multiple zip file manipulation operations at the same time, and is designed to use very little memory per instance – so your machine won't run out of memory when you're running dozens of instances at the same time.

The library also supports background processing. Background processing is a different concept than running multiple instances simultaneously. See the background processing property topic for details.

Any restrictions?

Yes, two small restrictions. First: two or more instances may not add files to, or create the same zip file located in the same folder. Second: each instance of the Xceed Zip control can only execute one process at a time. This makes sense, because if it was possible for a single instance to run more than one operation at a time, you would get the same event from two processes and wouldn't know for which process you are getting the event. Running multiple simultaneous operations involes having multiple instances, with each instance running its own operation.

How is it done?

If you're using forms in your application, you can place multiple Xceed Zip control objects on your form, call them XceedZip1, XceedZip2, XceedZip3, etc. Each control on the form is a separate instance of the library. Therefore, each instance's methods can be called simultaneously and can each process at the same time.

For example, place 4 Xceed Zip controls on a form, with 4 buttons, and 4 progress bars. For the click event of Button1, add code to zip up a file that refering to the XceedZip1 control. Set Button1's caption to the return value so you can know when it finished. In the XceedZip1 control's GlobalStatus event, set ProgressBar1's value to the nBytesPercent parameter provided to you by the GlobalStatus event. Do this 4 times, changine Button1 for Button2, XceedZip1 for XceedZip2, and ProgressBar1 for ProgressBar2. Make sure each button works on a different zip file! Now run your program, and click on each button one after the other. If you are zipping up enough files, each progress bar should be moving along at its own pace.

If you're using code, it's pretty much the same concept: Instantiate the control multiple times and instruct each instance to do whatever you want it to. In VB, to create multiple instances of the control, you can do this (this example can be generalized to more objects to n by using arrays):

```
' Make sure you added the Xceed Zip control to your object references. To do
    this, go to the Project menu and select the "References..." menu item,
    then checkmark the Xceed Zip Compression Library v4.0 entry in the list of
    components.
' Here we define the types for our variables
Dim WithEvents MyZipInstance1 As XceedZip
Dim WithEvents MyZipInstance2 As XceedZip
Dim WithEvents MyZipInstance3 As XceedZip
Dim WithEvents MyZipInstance4 As XceedZip
Dim ResultCode1 As xcdError
Dim ResultCode2 As xcdError
Dim ResultCode3 As xcdError
Dim ResultCode4 As xcdError
' Now we instantiate each variable with an Xceed Zip control
Set MyZipInstance1 = New XceedZip
Set MyZipInstance2 = New XceedZip
Set MyZipInstance3 = New XceedZip
Set MyZipInstance4 = New XceedZip
' Because we are running all 4 simultaneously from sequential code, we'll turn
    on background processing so that each of the following 4 lines is
    immediately executed, instead of first running the first line, waiting for
    it to complete, then running the second line, etc.
XceedZip1.BackgroundProcessing = True
```

```
XceedZip2.BackgroundProcessing = True
XceedZip3.BackgroundProcessing = True
XceedZip4.BackgroundProcessing = True
' Lets set our zip files to work with
XceedZip1.ZipFilename = "c:\temp\zipfile1.zip"
XceedZip2.ZipFilename = "c:\temp\zipfile2.zip"
XceedZip3.ZipFilename = "c:\temp\zipfile3.zip"
XceedZip4.ZipFilename = "c:\temp\zipfile4.zip"
' Which files to zip up
XceedZip1.FilesToProcess = "c:\temp\folder1\*"
XceedZip2.FilesToProcess = "c:\temp\folder2\*"
XceedZip3.FilesToProcess = "c:\temp\folder3\*"
XceedZip4.FilesToProcess = "c:\temp\folder4\*"
' Start all the zipping! (All resultcodes should = xecProcessStarted
ResultCode1 = XceedZip1.Zip
ResultCode2 = XceedZip1.Zip
ResultCode3 = XceedZip1.Zip
ResultCode4 = XceedZip1.Zip
' The rest of your code will continue executing, too. The only thing left is
    to write a handler for the ProcessCompleted event for each Xceed Zip
    instance, so you can get the xResult variable which will provide you with
    the real return value for each of the 4 zip calls above.
```

Formless use of the control

You can instantiate the ActiveX control without putting it on a form. To do this, make the control available in your development environment (this should already have been done if you followed the installation instructions) and simply create an instance of the object like you would any other variable. The object's type is "XceedZip".

Example for VB users:

```
' Make sure you added the Xceed Zip control to your object references. To do
    this, go to the Project menu and select the "References..." menu item,
    then checkmark the Xceed Zip Compression Library v4.0 entry in the list of
    components.
' Here we define the type for our variable
Dim WithEvents MyZipInstance1 As XceedZip
 ' Now we instantiate the variable with an Xceed Zip control instance
Set MyZipInstance1 = New XceedZip
 ' And now we can work as usual...
XceedZip1.ZipFilename = "c:\temp\zipfile1.zip"
XceedZip1.FilesToProcess = "c:\temp\folder1\*"
ResultCode1 = XceedZip1.Zip
...
```

GetZipFileInformation method example for VB

This example gets information for the zip file named "C:\TEST\MY ZIP FILE.ZIP", and puts the data in various variables that you can use. The example assumes you have placed a button, a listbox and an Xceed Zip control on a form, named Command1, List1 and XceedZip1 respectively.

```
Private Sub Command1_Click()
    Dim TotalFiles As Long
    Dim CompressedBytes As Long
    Dim UncompressedBytes As Long
    Dim CompressionRatio As Integer
    Dim IsZipFileSpanned As Boolean
    Dim ResultCode As xcdError
    ' Set the zip file name (the only property required to be set here)
    XceedZip1.ZipFilename = "c:\test\My zip file.zip"
    ' Get the information
    ResultCode = XceedZip1.GetZipFileInformation(TotalFiles, CompressedBytes,
    UncompressedBytes, CompressionRatio, IsZipFileSpanned)
    ' Check the return value
    If ResultCode <> xerSuccess Then
        List1.AddItem "Unsuccessful. Error #" + Str(ResultCode)
    Else
        List1.AddItem "Successfully retrieved the zip file's info"
        List1.AddItem "Total number of files: " + Str(TotalFiles)
        List1.AddItem "Compressed size: " + Str(CompressedBytes)
        List1.AddItem "Uncompressed size: " + Str(UncompressedBytes)
        List1.AddItem "Ratio: " + Str(CompressionRatio)+ "%"
        List1.AddItem "Spanned: " + Str(IsZipFileSpanned)
    End if
End Sub
```

GetZipFileInformation method example for Delphi

This example gets information for the zip file named "C:\TEST\MY ZIP FILE.ZIP", and puts the data in various variables that you can use. The example assumes you have placed a button, a listbox and an Xceed Zip control on a form, named Button1, List1 and XceedZip1 respectively.

```
procedure TForm1.Button1Click(Sender: TObject);
var
    TotalFiles : Integer;
    CompressedBytes : Integer;
    UncompressedBytes : Integer;
    CompressionRatio : SmallInt;
    IsZipFileSpanned : WordBool;
    ResultCode : xcdError;
begin
    ' Set the zip file name (the only property required to be set here)
    XceedZip1.ZipFilename := 'c:\test\My zip file.zip';
    ' Get the information
    ResultCode := XceedZip1.GetZipFileInformation(TotalFiles, CompressedBytes,
    UncompressedBytes, CompressionRatio, IsZipFileSpanned);
    ' Check the return value
    If ResultCode <> xerSuccess Then
        ListBox1.Items.Add("Unsuccessful. Error #" + Str(ResultCode)
    else
    begin
        ListBox1.Items.Add("Successfully got the zip file's info");
        ListBox1.Items.Add("Total files: " + IntToStr(TotalFiles));
        ListBox1.Items.Add("Comp. size: " + IntToStr(CompressedBytes));
        ListBox1.Items.Add("Size: " + IntToStr(UncompressedBytes));
        ListBox1.Items.Add("Ratio: " + IntToStr(CompressionRatio)+"%");
        ListBox1.Items.Add("Spanned: " + IntToStr(IsZipFileSpanned));
    end;
end;
```

GetErrorDescription method

Description

The GetErrorDescription method returns a text description of an xcdError, xcdWarning or xcdSkippingReason enumeration member.

This method is particularly useful in order to translate an error, warning or skipping reason code into english text in order to display a more detailed error than just the value of the error, warning or skipping reason code.

For example, to assign the description of the xerReadZipFile (505) error to a string, do this:

```
DescribeError = XceedZip1.GetErrorDescription(xvtError, 505);
or
DescribeError = XceedZip1.GetErrorDescription(xvtError, xerReadZipFile);
```

Declaration

```
Method GetErrorDescription(xType As xcdValueType, lValue As Long) As String
```

Parameters

xType

This parameter determines the category (error, warning or skipping reason) for the value of the lValue parameter.

lValue

The value of the enumeration member to get a text description of.

Return values

Returns the string containing the text description. If a non-existent member value is passed in the lValue parameter, an empty string will be returned.

xcdValueType enumeration

```
xcdValueType
{
      xvtError = 0,
      xvtWarning = 1,
      xvtSkippingReason = 2,
}
```

This enumeration is used by the GetErrorDescription method. Explanations for each value can be found in the GetErrorDescription method's topic.

About the CAB file

The XCEEDZIP.DLL is also provided as a digitally signed (with Microsoft Authenticode and VeriSign certificate) .CAB file. You can find the .CAB file in the BIN folder where you installed Xceed Zip v4.0.

Why a CAB file?

A .CAB file is known as a "cabinet" file. A cabinet file is a Microsoft-designed archive file format that's designed to pack multiple compressed files into a single file with a .CAB file extension. Cabinet files are used by some install programs (for example, Microsoft's Windows 98 "setup" program), and are also used to facilitate internet deployment of components and controls.

A .CAB file can contain an ActiveX control and all required DLLs and resources that the ActiveX control needs.

In the case of the XCEEDZIP.DLL, which is fully self-contained and does not require any other DLLs or resources, the .CAB file provided with this library is useful only to developers that are using the component on an HTML page. The .CAB file contains a compressed version of the XCEEDZIP.DLL, so it's smaller and reduces the download time for a browser that's receiving the HTML page and the control on the page. The .CAB file, like the XCEEDZIP.DLL file, is digitally signed with Microsoft Authenticode, so browsers won't complain about security issues.

Other remarks

The .CAB file contains an .INF file that Internet Explorer knows how to handle. This .INF file tells Internet Explorer how to install the correct version of the required DLLs in the appropriate location on the user's system.

When you are designing Active Server Page projects, there is no need for the .CAB file because the component stays on the server machine and is

About Migration

Thoughout the evolution of the Xceed Zip Compression Library, up until version 4.0, all 16 and 32-bit components shared the same programmatic interface. That means that as new components and features were added and Xceed Zip evolved from versions 1.0 to 3.5, all the controls offered by the library were programmed using the same properties, methods and events. When a new feature was added, and even when 32-bit support was introduced, Xceed Zip users required no modifications to their code.

For the Xceed Zip v4.0, which is an entirely new product based on a different programming paradigm and compression engine, an entirely new programming interface was designed and implemented. Xceed Zip v4.0 has some fundamental differences with Xceed Zip v3.5 in the way that it processes files, scans drives, fires events and handles the compressed data. Changing the programming interface has brought some definite advantages to the library. Xceed Zip v4.0's new interface offers developers a lot more flexibility and control, yet is more easier to use than ever. New users will get started faster, and veterans will enjoy the new flexibility offered by the library.

New Xceed Zip v4.0 ActiveX control follows the COM standard and all new versions and releases will remain compatible with the existing published interfaces. In no event shall the updating of the XCEEDZIP.DLL file affect existing applications.

The properties, methods and events used for Xceed Zip versions 1.0 to 3.5 have changed for Xceed Zip 4.0. Some have simply been renamed. Others no longer exist, and many work differently. If your existing applications that use Xceed Zip 1.0 to 3.5 wish to use the new features of Xceed Zip 4.0, some of your code must be modified. The "Migration" section of the Xceed Zip 4.0 manual will guide you through the changes you must make in order to get your code ready for using the new Xceed Zip 4.0 ActiveX control.

The code changes are outlined by the following topics:

- FilesToProcess property migration
- Other properties migration
- Other properties (SFX) migration
- Methods migration
- Events migration

FilesToProcess property migration

The most notable change when migrating your code to v4.0 concerns the FilesToProcess property. The following sections describe the changes to the FilesToProcess property that you must be aware of.

Wildcards

Wildcards specified in Xceed Zip v3.5's FilesToProcess property were used in order to determine which subfolders should be scanned when the Recurse property is set to True. For Xceed Zip version 4.0, wildcards are no longer used to determine which subfolders are scanned. You will also see later that the Recurse property has been renamed to ProcessSubfolders for Xceed Zip v4.0.

The new behavior for the FilesToProcess property has eliminated the need for the "Include only" character (>) that was often required when specifying wildcards to the FilesToProcess property. The new behavior is also more intuitive.

Example: with Xceed Zip v3.5 and before, if you wanted to zip all the .TXT files in the C:\DATA and all its subfolders, you had to use the following code:

```
XceedZip1.Recurse = True
XceedZip1.FilesToProcess = "C:\DATA\*" + vbCR + ">*.TXT"
```

This was necessary because if you only specified "C:\DATA*.TXT", Xceed Zip v3.5 would not scan subfolders because none of them would match "*.TXT". So, you had to specify that you wanted ALL files in C:\DATA, and that among those files, include only those that end with .TXT. With Xceed Zip v4.0, to accomplish the same task, use the following code:

```
XceedZip1.ProcessSubfolders = True
XceedZip1.FilesToProcess = "C:\DATA\*.TXT"
```

Excluded files

The FilesToProcess property no longer supports the "Exclude files" character (<) that could be specified in front of filenames specified to this property. You must use the FilesToExclude property now.

Other properties migration

Old property	Changes to note
ClearDisks	This property no longer exists. The DiskNotEmpty event is now triggered whenever a disk with already existing data is inserted. Setting it's xAction parameter will allow you to chose the action (including clearing the disk) to take.
Compression	Renamed to CompressionLevel
ExtractDirectory	Renamed to UnzipToFolder
FileCount	This property no longer exists. The GetZipFileInformation method allows you to retrieve the file count and other information from a zip file.
Recurse	Renamed to ProcessSubfolders. All subfolders encountered are now scanned, regardless of whether the subfolder's name matches a wildcard. (For more details see the FilesToProcess property migration topic)
FilesToProcess	Behavior has been modified. See the FilesToProcess property migration topic.
IncludeDirectoryEntries, IncludeVolumeLabel, IncludeHiddenFiles	These properties no longer exist. The RequiredFileAttributes and ExcludedFileAttributes properties are now used to determine which file attributes to include or exclude from the list of files to process. The default value of ExcludedFileAttributes property is xfaFolder + xfaVolume, which is the equivalent of: IncludeDirectoryEntries = False IncludeVolumeLabel = False IncludeHiddenFiles = True So if your existing code has any of these properties to False, make sure that the attribute is placed in ExcludedFileAttributes – otherwise, the corresponding file type will be included.
ModifiedDate	Renamed to MinDateToProcess.
MoveFiles	This property no longer exists. Use the FileStatus event to obtain the filenames of files you wish to delete, and if the Zip method's return value is xerSuccess, then you can delete the files.
MultidiskMode	Renamed to SpanMultipleDisks and changed behavior. The new property only applies when zipping, and is no longer necessary when unzipping, listing or testing.
Overwrite	This property no longer exists. The SkipIfExisting property now determines overwrite behavior. If your code had Overwrite = xowNever, it should now use SkipIfExisting = True. If you had Overwrite = xowAlways or xowAsk, it should now be SkipIfExisting = False. In the case of SkipIfExisting = False, the ReplacingFile event is triggered to allow you to prompt the end-user what to do. See the SkipIf* properties for more options.
Password	Renamed to EncryptionPassword.

Methods migration

Old method	Changes to note
Add	Renamed to Zip. No longer takes any parameters. Use the SkipIf* properties to reproduce the Add method's parameters. For example: Before: Add(xecAll) After: Zip Before: Add(xecFreshen) After: SkipIfNotExisting = True Zip Before: Add(xecUpdate) After: SkipIfExisting = True Zip
Delete	Renamed to RemoveFiles.
Extract	Renamed to Unzip. Use the SkipIf* properties to reproduce the Extract method's parameters. See the description for the Add method above for an example.
Fix	This method no longer exists. The new library will unzip everything it can from a corrupted zip file – there's no need to fix and unzip in two separate operations. However, the Convert method will allow you to repair a corrupted zip file if you still need this functionality.
GetFileInfo	This method no longer exists. You get information on a file by using the ListZipContents method and specifying the filename in the FilesToProcess property. The ListingFile event provides you with the info.
List	Renamed to ListZipContents.
MakeSelfExtracting	This method no longer exists. The Convert method allows you to (among other things) perform the same conversion operation. Example: Before: ZipFileName = "this.zip" SelfExtracting = True SfxBinary = "xcdsfx32.bin" MakeSelfExtracting After: ZipFileName = "this.zip" SfxBinaryModule = "xcdSfx32.bin" Convert("that.exe")
MemCompress, MemOriginalSize, MemUncompress, StringCompress, StringUncompress	The Xceed Compression control object now handles in-memory compression and decompression. Refer to that object for equivalent methods.
SfxResetDefaultStrings	Now divided into three methods called SfxResetButtons, SfxResetMessages and SfxResetStrings.
Test	Renamed to TestZipFile. You can now pass a parameter so you can control whether to test the zip structure only, or the zip structure and the integrity of the compressed data.
UpdateZipDate	This method no longer exists.

Events migration

All events have changed. It is recommended to reimplement events instead of renaming them. The parameters will all be properly set by your development language. Move your code as described here:

Old event	Changes to note
Adding	Move the status messages from this event to the FileStatus event. In the FileStatus event, you can consult the CurrentOperation property to customize your message depending on what you're doing. When the lBytesProcessed parameter = 0, the operation on that file is about to begin.
Deleting	Move status messages to the RemovingFile event.
Extracting	Move status messages to the FileStatus event in the same way as for the Adding event changes.
FileComment	Move file comment settings to the ZipPreprocessingFile event.
Fixing	This event no longer exists because the Fix method is no longer necessary.
GlobalStatus	The name and type of functionality has not changed, however you should make sure to reimplement the event, because parameter names and types have changed. See the GlobalStatus topic for details.
Listing	Move status messages to the ListingFile event.
Newdisk	Move the disk prompting dialogs or code to the InsertDisk event. Now you must set the bDiskInserted parameter to True when the user has confirmed inserting a disk, otherwise the process will stop with the xerInsertDiskAbort error.
PwdQuery	Move "file by file" custom password dialog box code to the ZipPreprocessingFile or UnzipPreprocessingFile events. Use the sPassword parameter to achieve the same functionality.
Rename	Move code that renames files to the ZipPreprocessingFile or UnzipPreprocessingFile events.
Replace	Move "replacing file" prompts to the ReplacingFile event. Keep in mind that you now have more information about the source and destination files. Set the bReplace parameter to False if you do not want a file replaced. Move "rename file" prompts to the ZipPreprocessingFile or UnzipPreprocessingFile events, and check the bExisting parameter to determine if a file should be renamed or not.
SkippingFile	The name and type of functionality has not changed, however you should make sure to reimplement the event, because parameter names and types have changed. See the SkippingFile topic for details.
Status	Move file status messages and progress bar updates to the FileStatus event.
Testing	Move status messages to the FileStatus event in the same way as for the Adding event changes.
Updating	Move status messages to the FileStatus event in the same way as for the Adding event changes. Keep in mind that the ZipPreprocessingFile and the UnzipPreprocessingFile events each have a parameter that tells you if the destination file exists or not.

Other properties (SFX) migration

Old property	Changes to note
SelfExtracting	This property no longer exists. If you enter a filename in the SfxBinaryModule property (formerly the SfxBinary property), it automatically implies SelfExtracting = True. Leaving the SfxBinaryModule property empty implies SelfExtracting = False.
SfxBinary	Renamed to SfxBinaryModule.
SfxConfigFile	Replaced by two methods called SfxSaveConfig and SfxLoadConfig that let you save and load the current state of all the Sfx-related properties.
SfxExtractDirectory	Renamed to SfxDefaultUnzipToFolder.
SfxIcon	Replaced by the SfxIconFilename property. A property of type Icon cannot be used in DCOM environments. You must now specify a filename.
SfxMessages, SfxPrompts	These two property arrays have been merged into one property array called SfxMessages. You can use the xcdMessages enumeration to get index values.
SfxPromptCreateDirectory, SfxPromptForDirectory, SfxPromptForPassword, SfxShowMessages, SfxShowProgress	These properties no longer exist. To be able to suppress one of these prompts or dialog boxes from being shown by the self-extracting zip file, simply leave the corresponding message in the SfxMessages property empty. For example, if you used SfxShowProgress = False, you would now use SfxMessages(xsmProgress) = "".
SfxOverwrite	Renamed to SfxExistingFileBehavior. Use xseAsk, xseSkip and xseOverwrite instead of xowAsk, xowNever and xowAlways.
SfxReadmePath	Renamed to SfxReadmeFile.
SfxRegisterExtensions	Renamed to SfxExtensionsToAssociate.
SfxRunExePath	Renamed to SfxExecuteAfter.
SfxStrings	The xssExtractingFile item was moved to the SfxMessages property array. You can now use the xcdSfxStgrings enumeration for indexes.
SfxStoredExtensions	This property no longer exists. If you want to change the compression method (stored or deflated), you can now do it on a file-by-file basis when the ZipPreprocessingFile event is triggered. Use it's xMethod parameter to control which files are stored and which are compressed.
TempPath	Renamed to TempFolder.
UsePaths	Renamed to PreservePaths.
ZipComment	Replaced by the ZipComment event. Use the ZipComment event's sComment parameter to set the desired comment while zipping, removing or converting a zip file. When unzipping or updating an existing zip file, the sComment parameter will contain the current zip comment.

16-bit support

The Xceed Zip Compression Library offers extensive 16-bit support.

Previous versions of the Xceed Zip Compression Library included 16-bit support in the form of a 16-bit Level-1 VBX (compatible with any language that can use a 16-bit VBX, such as Visual C++ 1.52 or Visual Basic 3.0 or 4.0) and a 16-bit Delphi 1 VCL.

These components have their own Windows help documentation, examples and samples.

Xceed Software will provide you with a 16-bit support package that contains the VBX and VCL components, free of charge, upon request. Simply write to sales@xceedsoft.com to request your 16-bit support package.

Self-Extractor Module

If you have purchased the Self-Extractor Module, the 16-bit components will be shipped with a full-featured Self-Extractor Module so you can create fully customized 16 and 32-bit self-extracting zip files from your 16-bit applications.

GetErrorDescription method

Description

The GetErrorDescription method returns a text description of an xcdCompressionError enumeration member.

This method is particularly useful in order to translate an error code into english text in order to display a more detailed error than just an error code value.

For example, to assign the description of the xceInvalidPassword (value 1004) error to a string, do this:

```
DescribeError = XceedCompression1.GetErrorDescription(1004);
or
DescribeError = XceedCompression1.GetErrorDescription(xceInvalidPassword);
```

Declaration

```
Method GetErrorDescription(lValue As Long) As String
```

Parameters

lValue

The value of the xcdCompressionError enumeration member to get a text description of.

Return values

Returns the string containing the text description. If a non-existent member value is passed in the lValue parameter, an empty string will be returned.