Zip Studio 2.0

Welcome!

Zip Studio 2.0 is a complete Zip\Unzip API for Windows 3.x. Zip Studio contains a set of DLLs, an easy to use VBX, 2 sample programs with source code, an enhanced MDI Zip Shell... and this help file. Of course Zip Studio is PKZip 2.04g compatible.

Overview

Why zipping?

Zip file compression methods are actually the most popular in the PC world. Why? Certainly because it' s a very powerful way to compress and it' s easy to use. Zipping\Unzipping is safe and fast, also, Windows programs become bigger and bigger so it' s quite compulsory to zip files to exchange them or to save costly hard disk space for example. With ZIP files, you can for example, save multiple files into one ZIP file, add comments, directories with an incredible compression rate (about, up to 80% saved space).

If you want to know more about zip files, you can download PKWare's products.

Why Zip Studio?

At the beginning, Zip was made to run under DOS (and other OS) as an utility EXE file. With Windows, maybe you don' t like to call "external" DOS EXE file. It' s slow, not really nice and, obviously, it doesn' t gain any advantage of the Windows environment. To hide this, you can find Windows sharewares which use a Windows interface to call a Dos Zip utility. If you' he already used these, maybe you think there are some natural limitations for these products: Batch configuration is not really easy, Dialog between interface and program is a few limited and when you zip you need to buy 2 products: the interface and the utility software...

Forget it! Now, with Zip Studio you can Zip\Unzip without concession. There are many ways to use our product: For example, you' r not a programmer but you use a well kown Word processor or a DataBase product. This one doesn' t allow you to compress your document and these, because they are really powerful (..), need an enormous disk space! Will you buy a new hard disk every 3 months? Sure, you won't. With Zip Studio you will integrate new Zip\ Unzip functionalities to your software. So you will save hard disk space and, if you are using diskette backup, one diskette will be big enough to make a backup or, if you prefer a tape backup, you will save a precious compression time. Second case, you are a programmer: Now, with Zip Studio you are free to read and write ZIP files in your works. Don' t you think it' s a good value?

Really powerful?

Zip Studio is distributed on 3 DLLs and a VBX. If you use the VBX, you will need all the 3 DLLs. To Zip, 2 DLLs are required, to Unzip, just one. These libraries contain about 30 high level functions. There are 3 main functions: One to zip, another to unzip and the last one to view a file in a zip. The unzip DLL also contains information functions. You can have a look at the ZDEMO example for these main functions and for notification messages treatment. The functions perform any required operations: for example, you don' t have to check if a file exist before calling them. Also, the VBX supports all DLLs functions and using this VBX instead of the API is far much easier.

Because Zip Studio is really easy to use, I also provide a BASIC include file and a BP7 interface. As a conclusion, if you wish to create a fully integrate Zip shell, you can use Zip Studio. Zip Studio includes advanced functions to test, repare, analyse, use comments, recurse directories..and much more...

References...

It' s a little too sson to give you a list of softwares built with the Zip Studio API. However you will probably see these next monthes some commercial softs or sharewares which use the Zip Studio, like the Huw MILLINGTON Windows Grep 1.5x...

Not registered version?

With the unregistered version, you can fully test Zip Studio. You will see Shareware principles reminders when you initialise the DLLs. You cannot specify directory in which you want to zip\unzip your files: When you zip, the ZIP file will be placed in the "WINDOWS\TEST" sub directory and when you unzip, all expanded files will also be placed in this directory. Distributing programs made with Zip Studio unregistered version is forbidden!

Support?

We offer a technical support for registered. users. You can join us thanks to COMPUSERVE (100333,27), Internet (hnet@dialup.francenet.fr) or, if you are really desperate you send us a mail (HEXANET - BP 385.16 - 75768 PARIS CEDEX 16 - FRANCE). If you ask for example, "How can I load a library with Visual Basic" or "Can I use your

product to zip a file?" you will never receive any reply! I hope you understand that the technical support is provided only for real questions regarding our product (and only Zip Studio) and if there is no answer in this help file.

Price?

Everybody can buy this shareware: It just costs 70$- or 390FF- with a Zip Shell licence. When you pay, you can freely distribute programs with Zip Studio functionalities. No royalties for the DLLs API nor for the VBX. If you buy the 'bundle' special offer and if you pay with a check drawn on an US Bank, you will also receive the last release on a diskette. Zip Studio 1.x registered users can buy the upgrade for just 20$.

First feet with Zip Studio

If you plan to use the VBX interface, see the VBX chapter instead...

Step 1 - Creating a Zip Studio project

Now, you certainly wish to create a sofware which calls Zip Studio functions. Call your development software, create a new project, select files and put ZDLL20A.DLL and ZDLL20B.DLL in your project path if you want to use Zip functions and do the same with UZDLL20.DLL if you want to do some unzipping.

Depending on the software (language) you use, you will link or not with "Importation Libraries". I only give LIB files for Visual C++, il you use BC++ for example, you will need to recreate these LIB files with IMPLIB or IMPLIBW. You must not make an import library for ZDLL20B.DLL. When your LIB files are correct, include them to your current project: Add ZDLL20A.LIB if you wish to ZIP and\or UZDLL20.LIB if you want to UNZIP.

If you are using a BASIC like language, you do not have to create or link LIB files.
Now your project is OK, call your main BAS, C, CPP, XXX file...

Step 2 - Declaring Zip Studio functions

Choose the file in which you want to initialise Zip Studio (generally this file is the main file of your project). If you are a C\C++ programmer, at the beginning of this file, just under the "#include <windows>", add "#include <zip.h>" and make sure that ZIP.H is in your INCLUDE specification directory. If you are a BASIC programmer, open ZIP.BAS and copy and paste interesting functions to your GLOBAL.BAS main file.

Step 3 - Calling the Zip Studio DLLs

Before you call all the functions you want, you must call ZipInit (if you want to use Zip functions) and \ or UnzipInit (if you want to use Unzip functions). You will do it at the beginning of your program. Afterwise, you can call all functions you want. When you will distribute your program, don' t forget to put required DLL in your package! That' s all. You can also use Setup Studio 2.2 (SSETUP22.ZIP) to create an installation program for your works. If you decide to buy the Zip Studio, don' t forget you can have a special offer for the 'bundle' (including Setup Studio 2.2)...

Zip functions

Zip functions allow you to create a zip file, to add or to delete files into this one and to put a comment into it. You can also repare this file if you think it has been damaged. All functions are placed in ZDLL20A.DLL and you must put ZDLL20A.DLL and ZDLL20B.DLL in your application path.

ZipInit	Init the ZDLL20A.DLL.
AddFileToZip	Add a file to a ZIP file. [2.0]
ZipDeleteFiles	Delete file(s) from a ZIP file.
ZipRepare	Recreate a ZIP file
ZipSetComment	Change the ZIP file comment
ZipSetLanguage	Fix the replace Dialog Box language
ZipSetReplaceText	Change the replace dialog box labels
ZipCancel	Stop the current Zip action [2.0]
ZipSetMode	Chose a compression mode [2.0]
ZipSplit	Split an existing Zip File [2.0]
ZipGetReplaceFlag	Get the user chosen overwrite mode

See also...

- Using Zip functions
- The first program
- Unzip functions
- Overview

ZipInit

void WINAPI ZipInit(LPCSTR szYourName, LPCSTR szPassword);

Description

This function must be called before any call to a ZIP function. When you register, I give you the 2 required passwords. If you are not a registered user, give "TEST" for <szYourName> and "" for <szPassword>. If you are not registered, this function will display a copyright reminder.

Example

```
WinMain( ... )
   {
   ZipInit( "TEST", "");
   ...
```

Required DLLs

ZDLL20A, ZDLL20B

See also...

UnzipInit
Registered version

Using ZIP functions

To help you to use ZIP functions, these are common ZIP questions and answers.

Speed up the process?

The best way to speed up the Zip processes is to use pattern instead of files name. Also, if you choose the background processing mode and if you use ZN_* notification messages, you must respond to these messages as soon as possible. When the ZIP process send a message to your main Window, it waits for the response, so if you make long treatment in your notification messages handler, the process will require up to 10x the original time! This is especially true with the ZN_ZIPPING message.

String size and Case?

If not specified, all paths, files name, mask... are limited to 180 characters. If you use more than 179 characters you may have a GPF, so check the user input. All functions are not case sensitive.

When do I need to call ZipInit function?

If you want to use ZIP functions (which are in ZDLL20A.DLL and ZDLL20B.DLL) you must call, before any call to these functions, ZipInit .Thanks to this function, you will specify if you' re or not a registered user. Generally, you will call this function at the beginning of your program to avoid code surcharge and the MessageBox if you are not a registered user. Also, if you want to you use UNZIP functions too, you need to call both UnzipInit and ZipInit.

Why Zipping process seems to be MONOTASK?

When you use AddFileToZip function, you must give a valid Window HANDLE to activate the backgound ZIP processing (multitask mode). If you give NULL, backgound processing mode is cancelled.

Why the ZipDeleteFiles doesn' t work?

When you use this function, you must specify paths if they are stored in the ZIP file. Else, files won' t be deleted.

What must I do with notification messages?

Notification messages are a powerful feature of Zip Studio. All functions which really process the ZIP file, will send you such messages. When you add an handler for these messages, it will work when you call ZIP functions (not only the AddFileToZip function).

Background processing?

When you use the background processing mode, don' t forget that the user can also close your application before the ZIP process is complete. That' s why you must prevent the user from closing your application during the process.

Storing directories?

In the Windows environment, we generally use full names instead of names without paths. When you choose to store paths in the ZIP file, this one will be relative to the ZIP path: For example, if the ZIP file is "C:\DIR1\MYZIP.ZIP" and the file to add, "C:\DIR1\DIR2\MYFILE.TXT", the stored full name will be "DIR2/MYFILE.TXT"...

AddFileToZip

int WINAPI AddFileToZip(LPCSTR szZipFile, LPCSTR szMask, int OverwriteMode, BOOL bStorePath, BOOL bRecurse, HWND hParentWnd, LPCSTR szPassword); [2.0]

Description

This is the main and only one ZIP function. <szZipFile> indicates the ZIP file to use or to create. If you are not a registered user, this ZIP file will be placed in "WINDOWS\TEST" directory. Use this function to add one or more file(s) specified with <szMask> to the ZIP file. <szMask> can use jokers like * or ? to make a filter, or specify a DOS filename. If you wish to receive notification messages, or, if you want to activate the background processing mode, you must give your main Window HWND as <hParentWnd>, elsewise you will specify NULL. <szPassword> is a password if you wish to encrypt the files. This password must not be longer than 79 characters.

<OverwriteMode> is one of these constants:

Name	*Value*	*Will*
OVERWRITE_ALWAYS	0	Always add files to ZIP file (default).
OVERWRITE_NEVER	1	Never overwrite files in ZIP file.
OVERWRITE_PROMPT	2	Ask the user if he wish or not to overwrite the file.
OVERWRITE_UPDATE	3	Overwrite if ZIP file version is older

<bStorePath> is one of these constants:

Name	*Value*	*Will*
STOREPATH_NO	0	Don' t keep the path.
STOREPATH_YES	1	Keep the file relative path.

<bRecurse> is one of these constants:

Name	*Value*	*Will*
RECURSE_NO	0	Do not recurse sub directories.
RECURSE_YES	1	Recurse sub directories

This function return one of the following value:

Name	*Value*	*Error level*
ZERROR_OK	0	OK
ZERROR_WARNING	1	Just a warning (not an error).
ZERROR_DESTFILE	2	Destfile (*.ZIP) error.
ZERROR_INTERNAL	3	Internal error.
ZERROR_FORMAT	4	Not a ZIP file.
ZERROR_NOMEM	5	Not enough memory to zip.
ZERROR_NOFILE	6	Nothing to do (can be just a warning).
ZERROR_NODLL	7	Can' t find ZDLL12B.DLL.
ZERROR_COMMENT	8	<not available here>
ZERROR_NOCRYPT	9	Can' t encrypt the files

If you give a valid <hParentWnd>, you will also receive these messages:

Name	*Value*	*Means*	*WParam*	*LParam*
ZN_ZIPPING	WM_USER + 46	Zipping a file.	Compressed size in Kb.	Current file name
ZN_FILEZIPPED	WM_USER + 45	File zipped	Compression rate.	Current file name
ZN_WRITING	WM_USER + 48	Replace Zip file	-	Zip file name
ZN_DELETING	WM_USER + 47	<na>	-	Current file name
ZN_REPARING	WM_USER + 49	<na>	Step: (1)Read\Write (2)Checking	(1)Current file name (2)Zip file name
ZN_COMPUTE	WM_USER +52	Count files before	Number of files	-
ZN_REPLACE	WM_USER +53	Replace dialog box displayed	TRUE if the Zip file version is newer.	Current file name
ZN_NETREAD	WM_USER +67	Read from net.	Rate	Current file name
ZN_NETWRITE	WM_USER +68	Write to net.	Rate	Current file name

Tips

1/ Using ZN_COMPUTE

It' s important to receive message because processing ZIP file before any action can take more than 30 seconds if your ZIP file contains more than 1000 files. ZN_COMPUTE notification message is sent to your <hParentWnd> before this action and tell you how many files will be treated.

2/ Make the user happy...

If you use the OVERWRITE_PROMPT mode, a dialog box will be shown to user, when necessary, and ask him to choose an overwrite mode (YES | NO | ALL | NEVER). Also, the user can change the overwrite mode. You have to read the new chosen overwrite mode with the ZipGetReplaceFlag function before you call again the main ZIP function.

3/ Bug in ZN_COMPUTE?...

We found some troubles with VC++, a CString, wsprintf, and the wParam value of ZN_COMPUTE. So if you use these, maybe you will find them too! I cant' get the right value for wParam when I use wsprintf (CString.GetBuffer(180), "%s ... %i", ..., (int)wParam);... When I use _itoa and CString Message += <ZN_COMPTE wParam value converted to a CString with _itoa>, it works well...so I 've used this method in ZDEMO program!

4/ No files lists...

This function, like other Zip functions, doesn' t support filenames or filters lists (like '*.TXT *.BAK') as Unzip functions do. If you give a list for the <szMask> parameter, this function won' t perform any treatment.

5/ Networks...

Zip Studio doesn' t work with diskless PCs and the network drives must be mapped from D: (eg R:)

6/ Encrypted files...

If you use a password, all the files in the current Zip file will be encrypted if they are not encrypted: So, you must add the encrypted files, then you can use 'normal' files.

Example

```
OnLaunchZipProcess(...)
   {
   char szZipFile[180];
   char szMask[180];
   int OverwriteMode = OVERWRITE_PROMPT, i;

   lstrcpy( szZipFile , CurrentZipFile );
   lstrcpy( szMask , FilesToZip );
   for ( i=0 ; i < 2 ; i++ )
      {
      lstrcpy( szMask , GetNextFileToZip(i) );
      if ( AddFileToZip( szZipFile, szMask, OverwriteMode, STOREPATH_NO,
      RECURSE_NO, hWnd ) > 0)
         {
         MessageBox( hWnd, "Error!","Zip Studio", MB_OK );
         return 0L;
         }
      OverwriteMode = ZipGetReplaceFlag(); // change the Overwritemode
      }
   }...
```

Required DLLs

ZDLL20A, ZDLL20B

See also...

ZipGetReplaceFlag
ZipDeleteFiles

ZipSetLanguage
ZipSetComment
IsFileUnzipable

ZipDeleteFiles

int WINAPI ZipDeleteFiles(LPCSTR szZipFile, LPCSTR szFiles, HWND hParentWnd);

Description

This function deletes files in a ZIP. <szZipFile> indicate the ZIP file, <szFiles> is the file name or the filter (with * and ?) for files you wish to delete, and <hParentWnd> is the notification messages receiver (your main Window) or NULL if you wish to cancel the background processing.

Returned value is one of the following:

Name	*Value*	*Error level*
ZERROR_OK	0	OK
ZERROR_WARNING	1	Just a warning (not an error).
ZERROR_DESTFILE	2	Destfile (*.ZIP) error.
ZERROR_INTERNAL	3	Internal error.
ZERROR_FORMAT	4	Not a ZIP file.
ZERROR_NOMEM	5	Not enough memory to zip.
ZERROR_NOFILE	6	Nothing to do (can be just a warning).
ZERROR_NODLL	7	Can' t find ZDLL12B.DLL.
ZERROR_COMMENT	8	<not available here>

Remark

If a ZIP file includes directories specification, you must also indicate a path in <szFiles>, for example, if ZIP files are "DIR1/FILE1.BAK", <szFiles> will be "DIR1\FILE1.BAK" or "DIR1*.BAK"

Example

```
if ( ZipDeleteFiles( "MYZIP.ZIP", "*.BAK", hWnd )>0 )
  MessageBox( hWnd, "Can' t delete *.BAK in MYZIP.ZIP","",MB_OK );
```

Required DLLs

ZDLL20A, ZDLL20B

See also...

AddFileToZip
ZipSetLanguage
ZipSetComment

ZipRepare

int WINAPI ZipRepare(LPCSTR szZipFile, HWND hParentWnd);

Description

The ZipRepare function tries to recreate a damaged ZIP file. You can also use IsThisFileAZipFile function to test the ZIP file before calling this function. <szZipFile> is the ZIP file name, <hParentWnd> is the notification messages receiver Window (your main Window) or NULL to cancel the background processing.

This function returns one of the following constants:

Name	*Value*	*Error level*
ZERROR_OK	0	OK
ZERROR_WARNING	1	Just a warning (not an error).
ZERROR_DESTFILE	2	Destfile (*.ZIP) error.
ZERROR_INTERNAL	3	Internal error.
ZERROR_FORMAT	4	Not a ZIP file.
ZERROR_NOMEM	5	Not enough memory to zip.
ZERROR_NOFILE	6	Nothing to do (can be just a warning).
ZERROR_NODLL	7	Can' t find ZDLL12B.DLL.
ZERROR_COMMENT	8	<not available here>

Example

```
if ( ZipRepare( "MYZIP.ZIP", hWnd) > 0 )
  MessageBox( hWnd, "MYZIP.ZIP is a desperate case!"
```

Required DLLs

ZDLL20A, ZDLL20B

See also...

IsThisFileAZipFile

ZipSetComment

int WINAPI ZipSetComment(LPCSTR szZipFile, LPCSTR szComment, HWND hParentWnd);

Description

This function adds, changes or, deletes the ZIP comment. ZIP comment is global to all files. A ZIP file can contain a comment up to 64 Kb, anyway, this function doesn' t accept comment bigger than 4 Kb. <szZipFile> is the ZIP file name, <szComment> the new comment or "" if you wish to delete the comment and, <hParentWnd> is the notification messages receiver Window (your main Window) or NULL to cancel the background processing. When you use this function, all the ZIP file is computed, so it' s a good idea to read the ZN_COMPUTE notification message value.

This function returns one of the following constants:

Name	*Value*	*Error level*
ZERROR_OK	0	OK
ZERROR_WARNING	1	Just a warning (not an error).
ZERROR_DESTFILE	2	Destfile (*.ZIP) error.
ZERROR_INTERNAL	3	Internal error.
ZERROR_FORMAT	4	Not a ZIP file.
ZERROR_NOMEM	5	Not enough memory to zip.
ZERROR_NOFILE	6	Nothing to do (can be just a warning).
ZERROR_NODLL	7	Can' t find ZDLL12B.DLL.
ZERROR_COMMENT	8	Comment is too big or NULL.

Tips

When you load a Zip file comment you will probably need to call <OemToAnsi> then you will replace "\r" with "\r\n". When you save (write) the comment you will replace "\r" with "\r\n".

Example

```
char* szComment = "My favourite comment...";
if ( ZipSetComment( "MYZIP.ZIP", szComment, hWnd )>0 )
  MessageBox( hWnd, "No comment set","Zip Studio", MB_OK );
```

Required DLLs

ZDLL20A, ZDLL20B

See also...

GetZipCommentLength
GetZipComment

ZipSetLanguage

BOOL WINAPI ZipSetLanguage(UINT cZipLanguage);

Description

Use this function if you wish to have the FRENCH, GERMAN or 'NONE' language interface for the replace dialog box. Default is set to English language (LANGUAGE_ENGLISH). Zip Studio 2.0 now supports GERMAN and 'NONE' languages. The 'NONE' language will remove any text from the replace dialog box, so, you will use it with the ZipSetReplaceText function (and when you receive the ZN_REPLACE notification message). Generally, you will call this function when you init the ZIP DLLs.This function returns TRUE if language is successfully set.

<cZipLanguage> is one of the following constants:

Name	*Value*	*Error level*
LANGUAGE_FRENCH	0	French mode.
LANGUAGE_ENGLISH	1	English mode.
LANGUAGE_GERMAN	2	German mode.
LANGUAGE_NONE	3	'NONE' mode.

Example

```
WinMain( ... )
   {
   ZipInit( "TEST", "");
   ZipSetLanguage( LANGUAGE_FRENCH );
   ...
```

Required DLLs

ZDLL20A, ZDLL20B

See also...

ZipInit
ZipSetReplaceText
Using Zip functions

ZipGetReplaceFlag

int WINAPI ZipGetReplaceFlag(void);

Description

When you choose the OVERWRITE_PROMPT mode, you need to give the user choice overwrite mode when you will call AddFileToZip again. This function returns the TRUE overwrite mode which is:

Name	*Value*	*Will*
OVERWRITE_ALWAYS	0	Always add files to ZIP file (default).
OVERWRITE_NEVER	1	Never overwrite files in ZIP file.
OVERWRITE_PROMPT	2	Ask the user if he wish or not overwrite the file.
OVERWRITE_UPDATE	3	Overwrite if ZIP file version is older

Example

```
if ( AddFileToZip( szZipFile, szMask, OverwriteMode, STOREPATH_NO,
RECURSE_NO, hWnd ) > 0)
   {
   MessageBox( hWnd, "Error!","Zip Studio", MB_OK );
   return 0L;
   }
OverwriteMode = ZipGetReplaceFlag(); // change the Overwritemode
}
```

Required DLLs

ZDLL20A, ZDLL20B

See also...

AddFileToZip

ZipSetReplaceText

BOOL WINAPI ZipSetReplaceText(LPCSTR szDialogTitle, LPCSTR szDialogText, LPCSTR szYesBtn, LPCSTR szNoBtn, LPCSTR szAlwaysBtn, LPCSTR szNeverBtn);

Description

This function will change the 'Replace' dialog box labels value. So you will use this function with the ZN_REPLACE notification message. If you wish to use a non standard language you will call this function with the informations the ZN_REPLACE will give to you and when you receive this message. As this function doesn' t perform any action if the dialog box is not here, in case you use Unzip functions too, you will call the UnzipSetReplaceText at the same time. <szDialogTitle> is the dialog box title, <szDialogText> is the replace main text ('the file ... is newer ...'), <szYesBtn> is the '&Yes' button text, <szNoBtn> is the '&No' button text, <szAlwaysBtn> is the '&Always' button text and <szNeverBtn> is the 'Ne&ver' button text. This function returns FALSE is the Zip replace dialog box is not available, TRUE in the other case.

Example

```
.../...
case ZN_REPLACE:
   {
   char szText[400];
   lstrcpy( szText, "The file  ");
   lstrcat( szText, (LPCSTR)lp ); // the file name
   lstrcat( szText, "  already exist...\n");
   if ( wp )
      lstrcat( szText, "The ZIP version is newer!");
   else
      lstrcat( szText, "The ZIP version is older.");
   // now call the both functions...
   ZipSetReplaceText( "Custom",szText,"&Yes","&No","&Always","Ne&ver");
   UnzipSetReplaceText( "Custom",szText,"&Yes","&No","&Always","Ne&ver");
   break;
   }
```

Required DLLs

ZDLL20A, ZDLL20B

See also...

ZN_REPLACE
AddFileToZip

Unzip functions

Unzip functions allow you to get informations or to unzip a zip file. You can also view a file in a ZIP file and, use general informations functions. All functions are placed in UZDLL20.DLL and you must put UZDLL20.DLL in your application path.

Initialisation

UnzipInit	Unzip initialisation function.

New Zip List information functions

GetZList	Build the Zip files list [2.0]
GetZipItem	Retrieve some infos for an item [2.0]
UnzipSortZList	Sort the Zip list once again [2.0]

Zip informations functions

IsFileInZip	Returns TRUE if a file is in a ZIP.
CountFileInZip	Returns the number of files in a ZIP.
GetFileNameFromZIP	Returns a filename of a file in a ZIP.
GetZFileOriginalSize	Returns the original size of a file in a ZIP.
GetZFileCompressedSize	Returns the compressed size of a file in a ZIP.
GetZFileDate	Returns the date of a file in a ZIP.
GetZFileTime	Returns the time of a file in a ZIP.
GetZCompressMethod	Returns the compression method for a file in a ZIP.
GetZFileIsDir	Returns TRUE if the file in a ZIP is a directory specification.
GetZFileIsEncrypted	Returns TRUE if the file in a ZIP is encrypted [2.0]
GetZFullInfos	Returns the full informations for a file in a ZIP.
IsThisFileAZipFile	Tests a ZIP file. [2.0]
IsFileUnzipable	TRUE if a filename of a ZIP is a DOS filename.
GetZipCommentLength	Returns the size of the ZIP comment.
GetZipComment	Retrieve the ZIP comment.

General informations functions

bDoesFileExist	Returns TRUE if a file exists.
GetShortFileName	Returns the file name without path.
GetPathFromFileName	Returns the path of a file name.
IsFileNameInFilter	TRUE if filename is in filter.

Extraction functions

ExtractZipFiles	Unzip a ZIP file.
GetQueryFlag	Returns the user's choice overwrite mode.
UnzipSetPassword	Set the password for the encrypted files [2.0]
SetAskPassword	Ask or not for a password [2.0]
UnzipCancel	Stop the current process as soon as possible [2.0].
ZipJoin	Join a multiple parts Zip file [2.0].

View function

ViewFileFromZip	View file(s) in a ZIP file.

Interface setting functions

UnzipSetMsgBoxTitle	Sets the replace dialog box title.
UnzipSetLanguage	Sets the replace dialog box language.
UnzipSetReplaceText	Change the replace dialog box labels.
UnzipSetPasswordText	Change the password dialog box labels. [2.0]
UnzipSetReceivingWindow	Sets the notification messages receiver Window.
UnzipSetBackgroundMode	En\Disable the background processing mode.

Required DLL

UZDLL20

See also...

- Using Unzip functions
- Zip functions
- Overview

Using Unzip functions

Speed up process?

The best way to speed up the UNZIP processes is to use patterns instead of files names. Also, if you choose the background processing mode and if you use ZN_* notification messages, you must respond to these messages as soon as possible. When the UNZIP process send a message to your main Window, it waits for the response, so if you make long treatment in your notification messages handler, the process will require up to 10x the original time! This is especially true with the ZN_EXPANDING message.

String size and Case?

If not specified, all paths, files name, mask... are limited to 180 characters. If you use more than 179 characters you may have a GPF, so check the user input. All functions are not case sensitive.

Must I use IsFileUnzipable before unzipping?

It' s not really necessary because the unzipping process tries to convert all names in DOS file name. Anyway, if you wish to be sure that all files with an UNIX like file name to be correctly unzipped, call this function. You can also use the IsFileUnzipable when you call a zip function in order to be sure the user choice is a valid DOS file name.

Background processing?

When you use the background processing mode, don' t forget that the user can also close your application before the UNZIP process is complete. That' s why you must prevent the user from closing your application during the process.

No ZN_COMPUTE?

There is no ZN_COMPUTE notification message for UNZIP functions. You can guess how much time process will require using the CountFileInZip function. Anyway, when you UNZIP, no extra time is required before the process.

Strings management

MFC users can use the String Studio 2.0 library for many unzip function (specially the GetZFileFullInfos) for string items extractions for instance. The String Studio 2.0 is available on Compuserve in the MSLANG Forum (its name is CSTR20.ZIP and its SWREG id is 3003).

UnzipInit

void WINAPI UnzipInit(LPCSTR szYourName, LPCSTR szPassword);

Description

Call this function before any call to an UNZIP function. When you register, We give you these two required passwords. If you are not a registered user give "TEST" for <szYourName> and "" for <szPassword>. Unregistered version will display a copyright reminder and when you will unzip files, they will be placed in the "WINDOWS\TEST" directory. Generally, you will call this function at the beginning of your program.

Example

```
WinMain( ... )
   {
   UnzipInit( "TEST", "");
   UnzipSetLanguage( LANGUAGE_FRENCH );
   ...
```

Required DLL

UZDLL20

See also...

ZipInit
Registered version

IsFileInZip

BOOL WINAPI IsFileInZip(LPCSTR szZIPFileName, LPCSTR szFileName);

Description

This function returns TRUE if the file <szFileName> is in the ZIP file specified with <szZIPFileName>. If your ZIP file contains path specification you must give them to <szFileName>. Obviously this function returns FALSE if the file isn't in the ZIP file.

Example

```
if ( IsFileInZip("MYZIP.ZIP", "READ.ME") )
  MessageBox( hWnd, "READ.ME found in MYZIP.ZIP","",MB_OK);
```

Required DLL

UZDLL20

See also...

GetFileNameFromZIP
IsThisFileAZipFile
IsFileUnzipable
GetZFileIsDir

CountFileInZip

int WINAPI CountFileInZip(LPCSTR szFileName);

Description

This function returns the number of files in a ZIP file or 0. <szFileName> specify the ZIP file name. If this function fails, -1 is returned.

Example

```
if ( CountFileInZip("MYZIP.ZIP") < 1)
   MessageBox( hWnd, "MYZIP is empty!","",MB_OK);
```

Required DLL

UZDLL20

See also...

- IsFileInZip
- GetFileNameFromZIP
- IsThisFileAZipFile
- IsFileUnzipable
- GetZFileFullInfos
- GetZFileIsDir

GetFileNameFromZIP

LPSTR WINAPI GetFileNameFromZIP(LPCSTR szFileName, UINT iFileNumber);

Description

This function returns the file name of a file in the ZIP file at position <iFileNumber>. <szFileName> is the ZIP file name and <iFileNumber> is the position (beginning at 1) of the file you want to retrieve the name. Returned name also includes relative path if available.

Example

```
char FirstFileInZip[180];
if ( CountFileInZip("MYZIP.ZIP") > 0 )
   lstrcpy( FirstFileInZip, GetFileNameFromZip("MYZIP.ZIP",1));
```

Required DLL

UZDLL20

See also...

CountFileInZip
IsThisFileAZipFile
IsFileUnzipable
GetZFileFullInfos
GetZFileIsDir

GetZFileOriginalSize

long WINAPI GetZFileOriginalSize(LPCSTR szZIPFileName, LPCSTR szFileName);

Description

Returns the original length of a file in a ZIP file. <szZIPFileName> is the ZIP file name and <szFileName> is the fully qualified name of the file you wish to get original size. The returned value is original file size in bytes, or 0L if function fails.

Example

```
char FirstFileInZip[180];
char Message[300];
long FirstFileOriginalSize = 0L;
if ( CountFileInZip("MYZIP.ZIP") > 0 )
   {
   lstrcpy( FirstFileInZip, GetFileNameFromZip("MYZIP.ZIP",1));
   FirstFileOriginalSize = GetZFileOriginalSize("MYZIP.ZIP",FirstFileInZip);
   wsprintf( Message, "%s original size is:%ld.", FirstFileInZip,
   FirstFileOriginalSize);
   MessageBox( hWnd, Message, "", MB_OK );
   }
```

Required DLL

UZDLL20

See also...

CountFileInZip
IsThisFileAZipFile
IsFileUnzipable
GetZFileCompressedSize
GetZFileFullInfos
GetZFileIsDir

GetZFileCompressedSize

long WINAPI GetZFileCompressedSize(LPCSTR szZIPFileName, LPCSTR szFileName);

Description

Returns the compressed length of a file in a ZIP file. <szZIPFileName> is the ZIP file name and <szFileName> is the fully qualified name of the file you wish to get compressed size. The returned value is compressed file size in bytes, or 0L if function fails.

Example

```
char FirstFileInZip[180];
char Message[300];
long FirstFileCompressedSize = 0L;
if ( CountFileInZip("MYZIP.ZIP") > 0 )
   {
   lstrcpy( FirstFileInZip, GetFileNameFromZip("MYZIP.ZIP",1));
   FirstFileCompressedSize =
   GetZFileCompressedSize("MYZIP.ZIP",FirstFileInZip);
   wsprintf( Message, "%s compressed size is:%ld.", FirstFileInZip,
   FirstFileCompressedSize);
   MessageBox( hWnd, Message, "", MB_OK );
   }
```

Required DLL

UZDLL20

See also...

CountFileInZip
IsThisFileAZipFile
IsFileUnzipable
GetZFileOriginalSize
GetZFileFullInfos
GetZFileIsDir

GetZFileDate

LPSTR WINAPI GetZFileDate(LPCSTR szZIPFileName, LPCSTR szFileName);

Description

Returns the last modification date for a file in the specified ZIP file. This value is returned as a string formatted depending on the current Windows configuration (specified in WIN.INI, "intl" section, sShortDate value). Items are always separated with the / character like 00/00/00. If function fails returned value is "" or "00/00/00" if no date is available for this file. <szZIPFileName> is the ZIP file name, <szFileName> is the fully qualified file name you want to get informations about.

About MFC CTime objects...

If you are a MFC user you can use our String Studio 2.0 library to translate the string into a date (a CTime object) for instance. The String Studio 2.0 is available on Compuserve in the MSLANG Forum (its name is CSTR20.ZIP and its SWREG id is 3003).

Example

```
char FirstFileInZip[180];
char Message[300];
char LastDate[15];
if ( CountFileInZip("MYZIP.ZIP") > 0 )
   {
   lstrcpy( FirstFileInZip, GetFileNameFromZip("MYZIP.ZIP",1));
   lstrcpy( LastDate, GetZFileDate("MYZIP.ZIP",FirstFileInZip));
   wsprintf( Message, "%s last modification date is:%s.", FirstFileInZip,
   LastDate);
   MessageBox( hWnd, Message, "", MB_OK );
   }
```

Required DLL

UZDLL20

See also...

CountFileInZip
IsThisFileAZipFile
IsFileUnzipable
GetZFileTime
GetZFileFullInfos
GetZFileIsDir

GetZFileTime

LPSTR WINAPI GetZFileTime(LPCSTR szZIPFileName, LPCSTR szFileName);

Description

Returns the last modification time for a file in the specified ZIP file. This value is returned as a string formatted depending of the current Windows configuration (specified in WIN.INI, "intl" section, iTime value). Items are always separated with the : character like 00:00 or 00:00am or 00:00pm. If function fails returned value is "" or "00:00" if no time is available for this file. <szZIPFileName> is the ZIP file name, <szFileName> is the fully qualified file name you want to get informations about.

About MFC CTime objects...

If you are a MFC user you can use our String Studio 2.0 library to translate the string into a date (a CTime object) for instance. The String Studio 2.0 is available on Compuserve in the MSLANG Forum (its name is CSTR20.ZIP and its SWREG id is 3003).

Example

```
char FirstFileInZip[180];
char Message[300];
char LastTime[15];
if ( CountFileInZip("MYZIP.ZIP") > 0 )
   {
   lstrcpy( FirstFileInZip, GetFileNameFromZip("MYZIP.ZIP",1));
   lstrcpy( LastTime, GetZFileTime("MYZIP.ZIP",FirstFileInZip));
   wsprintf( Message, "%s last modification time is:%s.", FirstFileInZip,
   LastTime);
   MessageBox( hWnd, Message, "", MB_OK );
   }
```

Required DLL

UZDLL20

See also...

CountFileInZip
IsThisFileAZipFile
IsFileUnzipable
GetZFileDate
GetZFileFullInfos
GetZFileIsDir

GetZCompressMethod

UINT WINAPI GetZCompressMethod(LPCSTR szZipFileName, LPCSTR szFileName);

Description

This function returns the compression method used for <szFileName> in ZIP file <szZipFileName>.<szFileName> is the fully qualified name of the file you want to get informations about.

Returned value is one of the following:

Name	*Value*	*Means*
ZMETHOD_STORED	0	File is stored (Zip 1.0 compatable)
ZMETHOD_SHRUNK	1	File is Shrunked
ZMETHOD_REDUCE1	2	File is reduced (level 1)
ZMETHOD_REDUCE2	3	File is reduced (level 2)
ZMETHOD_REDUCE3	4	File is reduced (level 3)
ZMETHOD_REDUCE4	5	File is reduced (level 4)
ZMETHOD_IMPLODE	6	File is imploded
ZMETHOD_TOKEN	7	File is tokened
ZMETHOD_DEFLATE	8	File is deflated (default)
ZMETHOD_UNKNOWN	9	Not compatable with Zip Studio.
ZMETHOD_ERROR	10	ZIP file damaged

Example

```
char FirstFileInZip[180];
if ( CountFileInZip("MYZIP.ZIP") > 0 )
   {
   lstrcpy( FirstFileInZip, GetFileNameFromZip("MYZIP.ZIP",1));
   if ( GetZCompressMethod("MYZIP.ZIP",FirstFileInZip)> 8)
     MessageBox( hWnd, "Can't unzip the first file!","MYZIP.ZIP", MB_OK);
   }
```

Required DLL

UZDLL20

See also...

CountFileInZip
IsThisFileAZipFile
IsFileUnzipable
GetZFileOriginalSize
GetZFileCompressedSize
GetZFileFullInfos
GetZFileIsDir

GetZFileIsDir

BOOL WINAPI GetZFullInfos(LPCSTR szZIPFileName, LPCSTR szFileName);

Description

This function returns TRUE is the file name <szFileName> in the Zip file <szZIPFileName> is a directory specification and not a true file. This function probably won' t work with UNIX file names.

Example

```
if ( GetZFileIsDir("MYZIP.ZIP",szFirstFileInZip))
  MessageBox( hWnd, szFirstFileInZip,"MYZIP.ZIP - Directory:", MB_OK);
```

Required DLL

UZDLL20

See also...

CountFileInZip
IsThisFileAZipFile
IsFileUnzipable
GetZFileOriginalSize
GetZFileCompressedSize
GetZFileFullInfos

GetZFullInfos

BOOL WINAPI GetZFullInfos(LPCSTR szZipFileName, LPCSTR szFileName, LPSTR szResult);

Description

This function fills <szResult> with the informations for the file <szFileName> from the ZIP file <szZipFileName>. <szResult> must be a valid buffer for 456 characters. The ';' character will be used to separate each value in this order: The current file name (same as <szFileName>), the Original size in bytes, the compressed size, the file date, the file time and the compression method. For instance, <szResult> coul be 'MYFILE.TXT;1254;984;01/21/94;04:12pm;8'. String Studio 1.2 (CSTR12.ZIP) can help you to make the items parsing if you are a MFC user. This function returns TRUE if the data were read, FALSE in the other case.

About MFC CTime objects...

If you are a MFC user you can use our String Studio 2.0 library to extract some items for instance. The String Studio 2.0 is available on Compuserve in the MSLANG Forum (its name is CSTR20.ZIP and its SWREG id is 3003).

Example

```
char szTest[456];
  if ( GetZFullInfos("MYZIP.ZIP",FirstFileInZip, szTest))
    MessageBox( hWnd, szTest,"MYZIP.ZIP - Infos", MB_OK);
```

Required DLL

UZDLL20

See also...

CountFileInZip
IsThisFileAZipFile
IsFileUnzipable
GetZFileOriginalSize
GetZFileCompressedSize
GetZFileIsDir

IsThisFileAZipFile

BOOL WINAPI IsThisFileAZipFile(LPCSTR szFileName, BOOL bQuickTest); [2.0]

Description

This function returns TRUE if <szFileName> doesn' t seem to be damaged, FALSE in the other case. <szFileName> is the ZIP file name. If you think the ZIP file is corrupted, you can use the ZipRepare ZIP function to recreate the ZIP file. If <bQuicktest> is set to TRUE, the function won' t check every files. If set to FALSE, the function will send a ZN_TEST notification message for all the tested files.

Example

```
if ( !IsThisFileAZipFile("MYZIP.ZIP", TRUE) )
   {
   if ( ZipRepare("MYZIP.ZIP", hWnd) > 0 )
     MessageBox( hWnd, "MYZIP.ZIP is a desperate case!","",MB_OK);
   }
```

Required DLL

UZDLL20

See also...

ZipRepare
bDoesFileExist

IsFileUnzipable

BOOL WINAPI IsFileUnzipable(LPCSTR szFileName);

Description
This function returns TRUE if <szFileName> is a DOS compatable file name, FALSE if not.

About MFC CTime objects...
If you are a MFC user you can use our String Studio 2.0 library to use and to check files names or patterns for instance. The String Studio 2.0 is available on Compuserve in the MSLANG Forum (its name is CSTR20.ZIP and its SWREG id is 3003).

Example
```
char FirstFileInZip[180];
if ( CountFileInZip("MYZIP.ZIP") > 0 )
  {
  lstrcpy( FirstFileInZip, GetFileNameFromZip("MYZIP.ZIP",1));
  if ( !IsFileUnzipable(FirstFileInZip))
    MessageBox( hWnd, "Can't unzip the first file!","MYZIP.ZIP", MB_OK);
  }
```

Required DLL
UZDLL20

See also...
GetShortFileName
GetPathFromFileName
IsFileNameInFilter

GetZipCommentLength

UINT WINAPI GetZipCommentLength(LPCSTR szFileName);

Description

Use this function to retrieve the length of the ZIP global comment. The length of this one can be as big as 64 Kb. <szFileName> is the ZIP file name. This function returns the length of the global ZIP comment (not including the last void character) or 0 if there is no ZIP global comment.

Example

```
HANDLE hComment;
char* szCommentBuffer;
UINT CommentLength = 0;
if ( (CommentLength = GetZipCommentLength("MYZIP.ZIP")) > 0 )
   {
   hComment = GlobalAlloc( GHND, (long)(CommentLength+2));
   szCommentBuffer = (LPSTR)GlobalLock( hComment );
   lstrcpy( szCommentBuffer, "");
   GetZipComment( "MYZIP.ZIP", szCommentBuffer );
   if ( lstrlen( szCommentBuffer )> 0 )
     MessageBox( hWnd, szCommentBuffer, "MYZIP.ZIP Comment", MB_OK );
   else
     MessageBox( hWnd, "Can' t get it!, "MYZIP.ZIP Comment", MB_OK );
   }
```

Required DLL

UZDLL20

See also...

GetZipComment
IsThisFileAZipFile
ZipSetComment

GetZipComment

BOOL WINAPI GetZipComment(LPCSTR szFileName, LPSTR szBuffer);

Description

Use this function to read the ZIP file global comment. To use this function, call GetZipCommentLength before. <szFileName> is the ZIP file name, <szBuffer> will be filled with the ZIP comment or "" if there is no available comment. The returned value is TRUE if comment has been read, FALSE if not. You can use <OemToAnsi> when you load a Zip comment and replace "\r" with "\r\n".

Example

```
HANDLE hComment;
char* szCommentBuffer;
UINT CommentLength = 0;
if ( (CommentLength = GetZipCommentLength("MYZIP.ZIP")) > 0 )
   {
   hComment = GlobalAlloc( GHND, (long)(CommentLength+2));
   szCommentBuffer = (LPSTR)GlobalLock( hComment );
   lstrcpy( szCommentBuffer, "");
   GetZipComment( "MYZIP.ZIP", szCommentBuffer );
   if ( lstrlen( szCommentBuffer )> 0 )
     MessageBox( hWnd, szCommentBuffer, "MYZIP.ZIP Comment", MB_OK );
   else
     MessageBox( hWnd, "Can' t get it!, "MYZIP.ZIP Comment", MB_OK );
   }
```

Required DLL

UZDLL20

See also...

GetZipCommentLength
IsThisFileAZipFile
ZipSetComment

bDoesFileExist

BOOL WINAPI bDoesFileExist(LPCSTR szFileName);

Description

This function returns TRUE if <szFileName> exists, FALSE in the other case.

Example

```
if ( !bDoesFileExist("MYZIP.ZIP"))
  MessageBox(hWnd,"MYZIP.ZIP is not in the current directory!","",MB_OK);
```

Required DLL

UZDLL20

See also...

GetShortFileName
GetPathFromFileName
IsFileNameInFilter
IsThisFileAZipFile

GetShortFileName

LPSTR WINAPI GetShortFileName(LPCSTR szFileName);

Description

This function returns the file name without directory specification for <szFileName>. If <szFileName> doesn' t specify any directory its full value will be returned. You can use jokers (* or ?) and / or \ to specify paths for <szFileName>.

Example

```
char DOSFileName[15];
char DOSPath[180];
lstrcpy( DOSFileName, GetShortFileName("C:\\ZIP\\MYZIP.ZIP"));
lstrcpy( DOSPath, GetPathFromFileName("C:\\ZIP\\MYZIP.ZIP"));
MessageBox( hWnd, DOSFileName, "Name", MB_OK );
MessageBox( hWnd, DOSPath, "Path", MB_OK );
```

Required DLL

UZDLL20

See also...

GetPathFromFileName
IsFileNameInFilter
IsThisFileAZipFile
bDoesFileExist

GetPathFromFileName

LPSTR WINAPI GetPathFromFileName(LPCSTR szFileName);

Description

This function returns the path for <szFileName>. If <szFileName> doesn' t specify any directory "" will be returned. If there is a directory specification, this one will end with "\". You can use jokers (* or ?) and / or \ to specify paths for <szFileName>. Returned path can' t be longuer than 180 characters and ends with "\".

Example

```
char DOSFileName[15];
char DOSPath[180];
lstrcpy( DOSFileName, GetShortFileName("C:\\ZIP\\MYZIP.ZIP"));
lstrcpy( DOSPath, GetPathFromFileName("C:\\ZIP\\MYZIP.ZIP"));
MessageBox( hWnd, DOSFileName, "Name", MB_OK );
MessageBox( hWnd, DOSPath, "Path", MB_OK );
```

Required DLL

UZDLL20

See also...

GetShortFileName
IsFileNameInFilter
IsThisFileAZipFile
bDoesFileExist

IsFileNameInFilter

BOOL WINAPI IsFileNameInFilter(LPCSTR szFileName, LPCSTR szMask);

Description

This function returns TRUE if <szFileName> matches <szMask>. <szMask> can be a file name or a mask using * or ?. Also you can specify multiple masks if you separate them with a space (" "): For example, "*.BAK MYFILE.TXT" will be a valid mask. Also, you can specify directories for <szFileName> or <szMask> but they will be ignored.

Example

This example will unzip, at best, only the first file of the ZIP file, if this file matches the user specified filter

```
char FirstFileInZip[180];
char UserMask[180];
...
lstrcpy( UserMask, GetUserChoosenFilter());
...
if ( CountFileInZip("MYZIP.ZIP") > 0 )
   {
   lstrcpy( FirstFileInZip, GetFileNameFromZip("MYZIP.ZIP",1));
   if ( IsFileNameInFiler( FirstFileInZip, UserMask ))
     ExtractZipFiles( "MYZIP.ZIP",FirstFileInZip,"",OVERWRITE_TRUE,CREATEDIR_F
     ALSE);
   ...
   }
```

Required DLL

UZDLL20

See also...

GetPathFromFileName
GetShortFileName
IsFileUnzipable
IsThisFileAZipFile
bDoesFileExist

ExtractZipFiles

UINT WINAPI ExtractZipFiles(LPCSTR szFileName, LPCSTR szMask, LPCSTR szDestDir, UINT bOverwrite, BOOL bCreateDir);

Description

This is the main UNZIP function. This one will extract files specified by <szMask> of ZIP file <szFileName> into <szDestDir> destination directory if specified and, will recreate directories if they are specified in the ZIP file and if you choose the CREATEDIR_TRUE mode. If you aren' t a registered user, all unzipped files will be placed in the "WINDOWS\TEST" directory.

<szFileName> is the ZIP file name, <szMask> is the file(s) to UNZIP specification and can be a file name, a mask using jokers (* or ?) or a list of file names or masks separated with a space (like "*.BAK *.TXT"). If you don' t specified any <szMask>, "*.*" will be used. You don' t have to specify any path for <szMask>. <szDestDir> is the name of the destination directory. <szDestDir> must be specified (if you are a registered user) and must not end with the '\" character.

To use encrypted Zip files you must call UnzipSetPassword and to use multiple parts Zip files, you must join them with ZipJoin before you can call this routine.

<bOverwrite> is the overwrite mode and can be:

Name	*Value*	*Means*
OVERWRITE_FALSE	0	Don' t UNZIP files if they already exist.
OVERWRITE_TRUE	1	Always UNZIP files.
OVERWRITE_QUERY	2	Ask the user if he wish to overwrite a file.

<bCreateDir> can be one of of the following:

Name	*Value*	*Means*
CREATEDIR_FALSE	0	Do not recreate dir if stored in the ZIP file.
CREATEDIR_TRUE	1	Recreate directories stored in the ZIP file.

This function returns one of the following value:

Name	*Value*	*Means*
ZEXTRACT_OK	0	OK.
ZEXTRACT_INTERNALERROR	1	Fails to unzip files.
ZEXTRACT_FILENOTFOUND	2	Can' t find a file.
ZEXTRACT_CORRUPTED	3	Not a ZIP file.
ZEXTRACT_EMPTY	4	Nothing to do (can be just a warning).
ZEXTRACT_ERRORINZIPFILE	5	ZIP file corrupted.
ZEXTRACT_NOMEM	6	Not enough memory to UNZIP.
ZEXTRACT_DISKFULL	8	Disk is full!
ZEXTRACT_WARNING	10	Just a warning.

If you want to receive process notification messages, you must use the UnzipSetReceingWindow to activate it. So you will receive these messages:

Name	*Value*	*Means*	*WParam*	*LParam*
ZN_OPENFILE	WM_USER + 38	Open a file.	TRUE if OK.	Current filename
ZN_EXPANDING	WM_USER + 39	Unzipping a file.	Current fill rate.(0..100).	Current filename
ZN_CLOSEFILE	WM_USER + 40	Closing a file.	TRUE if OK.	Current filename
ZN_REPLACE	WM_USER + 53	Replace dialog box is displayed.	TRUE if the ZIP version is newer.	Current filename
ZN_NEWPASSWORD [2.0]	WM_USER + 65	New password required.	-	the file name, the tab character and the current password.
ZN_TEST [2.0]	WM_USER + 69	Send by <IsThisFile...>	TRUE if the file is ok	checked file name.

If you use the OVERWRITE_QUERY mode, a dialog box will be shown to user, when necessary, and ask him to choose an overwrite mode (YES | NO | ALL | NEVER). Also, the user can change the overwrite mode. You have to read the new chosen overwrite mode with the GetQueryFlag function before you call again the main UNZIP function.

Example

This first example will unzip, at best, only the first and the last files of the ZIP file, if these files match the user specified filter.

```
char FirstFileInZip[180];
char LastFileInZip[180];
char UserMask[180];
int LastFilePos;
int OverwriteMode = OVERWRITE_QUERY;
...
lstrcpy( UserMask, GetUserChoosenFilter());
...
if ( (LastFilePos = CountFileInZip("MYZIP.ZIP")) > 0 )
  {
  lstrcpy( FirstFileInZip, GetFileNameFromZip("MYZIP.ZIP",1));
  if ( IsFileNameInFilter( FirstFileInZip, UserMask ))
    ExtractZipFile( "MYZIP.ZIP",FirstFileInZip,"",OverwriteMode,CREATEDIR_FAL
    SE);
  ...
  }
```

Now have a look at the overwrite mode before unzipping the last file

```
if ( LastFilePos>1 )
  {
  lstrcpy( LastFileInZip, GetFileNameFromZip("MYZIP.ZIP",LastFilePos));
  if ( IsFileNameInFilter( LastFileInZip, UserMask ))
    {
    OverwriteMode = GetQueryFlag();
    ExtractZipFile( "MYZIP.ZIP",LastFileInZip,"",OverwiteMode,CREATEDIR_FALSE
    );
    ...
    }
```

Anyway, we can rewrite this previous example like this, which dooesn' t call the GetQueryFlag function:

```
char FirstFileInZip[180];
char LastFileInZip[180];
char OurNewMask[360];
char UserMask[180];
int LastFilePos;
int OverwriteMode = OVERWRITE_QUERY;
...
lstrcpy( UserMask, GetUserChoosenFilter());
lstrcpy( LastFileInZip, "");
...
if ( (LastFilePos = CountFileInZip("MYZIP.ZIP")) > 0 )
  {
  lstrcpy( FirstFileInZip, GetFileNameFromZip("MYZIP.ZIP",1));
  lstrcpy( OurNewMask, FirstFileInZip);
    if ( LastFilePos > 1 )
    {
    lstrcat( OurNewMask, " ");
    lstrcpy( LastFileInZip, GetFileNameFromZip("MYZIP.ZIP",LastFilePos));
    lstrcat( OurNewMask, LastFileInZip );
    }
```

```
if ( (IsFileNameInFilter( FirstFileInZip, UserMask )) &&
(IsFileNameInFilter( LastFileInZip, UserMask )))
  ExtractZipFile( "MYZIP.ZIP",OurNewMask,"",OverwriteMode,CREATEDIR_FALSE);
else if ( (IsFileNameInFilter( FirstFileInZip, UserMask )) && !
(IsFileNameInFilter( LastFileInZip, UserMask )))
  ExtractZipFile( "MYZIP.ZIP",FirstFileInZip,"",OverwriteMode,CREATEDIR_FAL
  SE);
else if ( !(IsFileNameInFilter( FirstFileInZip, UserMask )) &&
(IsFileNameInFilter( LastFileInZip, UserMask )))
  ExtractZipFile( "MYZIP.ZIP",LastFileInZip,"",OverwriteMode,CREATEDIR_FALS
  E);
...
```

Required DLL

UZDLL20

See also...

GetQueryFlag
UnzipSetMsgBoxTitle
UnzipSetLanguage
UnzipSetReplaceText
UnzipSetReceivingWindow
UnzipSetBackgroundMode
IsFileUnzipable
IsThisFileAZipFile
IsFileNameInFilter
ZipRepare
AddFileToZip

GetQueryFlag

int WINAPI GetQueryFlag(void);

Description

This function returns the last Overwrite mode for unzipping. It' s useful when we choose the OVERWRITE_QUERY mode. You don' t have to read this value when you don' t use the <Overwrite Query> mode. Returned value is one of the following:

Name	*Value*	*Means*
OVERWRITE_FALSE	0	Don' t UNZIP files if they already exist.
OVERWRITE_TRUE	1	Always UNZIP files.
OVERWRITE_QUERY	2	Ask the user if he wish to overwrite a file.

Example

This first example will unzip, at best, only the first and the last files of the ZIP file, if these files match the user specified filter.

```
char FirstFileInZip[180];
char LastFileInZip[180];
char UserMask[180];
int LastFilePos;
int OverwriteMode = OVERWRITE_QUERY;
...
lstrcpy( UserMask, GetUserChosenFilter());
...
if ( (LastFilePos = CountFileInZip("MYZIP.ZIP")) > 0 )
   {
   lstrcpy( FirstFileInZip, GetFileNameFromZip("MYZIP.ZIP",1));
   if ( IsFileNameInFilter( FirstFileInZip, UserMask ))
      ExtractZipFile( "MYZIP.ZIP",FirstFileInZip,"",OverwriteMode,CREATEDIR_FAL
      SE);
   ...
   }
```

Now have a look at the overwrite mode before unzipping the last file

```
if ( LastFilePos>1 )
   {
   lstrcpy( LastFileInZip, GetFileNameFromZip("MYZIP.ZIP",LastFilePos));
   if ( IsFileNameInFilter( LastFileInZip, UserMask ))
      {
      OverwriteMode = GetQueryFlag();
      ExtractZipFile( "MYZIP.ZIP",LastFileInZip,"",OverwiteMode,CREATEDIR_FALSE
      );
      ...
      }
```

Required DLL

UZDLL20

See also...

ExtractZipFile

ViewFileFromZip

BOOL WINAPI ViewFileFromZip(LPCSTR szZipFile, LPCSTR szMask, BOOL bTextOnly);

Description

Sometimes you don' t want to UNZIP a file directly... For example, you' ve already got an unzipped file version in your ZIP file directory and you don' t know what version is the good one. Another example: You' ve bought a new CD-ROM and you' re in a hurry to have a look at its ZIP files. Sure, you don' t want to get bored in copying these files to your hard disk, unzipping them and finally watch them.

The best solution is to view them before you decide to unzip them or not. My view function do this task and use the ready to use File Manager data and programs associations. You can also choose to bypass these associations if you set <bTextOnly> to TRUE. In this case, every files will be displayed with the TXT associated program (NOTEPAD.EXE by default).

To call this function, give the ZIP file name for <szZipFile>, one or more filename or mask(s) using jokers (* or ?) and separated with a space (" ") to <szMask>.

This function will call the UNZIP process (so you will receive notification messages) but will create temporary files in the WINDOWS directory instead of normal files. When the file is completely filled, the EXE, PIF, COM or BAT program is called with this temporary file name (~UNZIP*.TMP) and when this program is ready, the function will destroy the temporary file and will process the next file. If the file is encrypted, the Zip Studio Shell will display the password dialog box.

Notification messages this function send, are described in ExtractZipFile.

This function returns TRUE if all files have been displayed, FALSE if one or more files can' t be displayed.

Example

This example display the first file of MYZIP.ZIP

```
char FirstFileInZip[180];
if ( CountFileInZip("MYZIP.ZIP") > 0 )
   {
   lstrcpy( FirstFileInZip, GetFileNameFromZip("MYZIP.ZIP",1));
   ViewFileFromZip( "MYZIP.ZIP", FirstFileInZip, FALSE );
   ...
   }
```

Required DLL

UZDLL20

See also...

ExtractZipFile
IsThisFileAZipFile

UnzipSetMsgBoxTitle

BOOL WINAPI UnzipSetMsgBoxTitle(LPCSTR szNewTitle);

Description

This function changes the title of the UNZIP replace dialog box (default is "Replace..."). <szNewTitle> is the new title of this dialog box and can' t be longuer than 160 characters. This function returns TRUE if the new title is set, FALSE if not. This function is not available for unregistered users. Now you will use the UnzipSetReplaceText instead of this function.

Example

```
WinMain( ... )
   {
   UnzipInit( "TEST", "");
   UnzipSetMsgBoxTitle("Overwrite?");
   UnzipSetLanguage( LANGUAGE_FRENCH );
...
```

Required DLL

UZDLL20

See also...

- UnzipSetLanguage
- UnzipSetReceivingWindow
- UnzipSetBackgroundMode
- UnzipInit
- UnzipSetReplaceText

UnzipSetLanguage

void WINAPI UnzipSetLanguage(UINT iLanguage);

Description

This function changes the language used by UNZIP replace dialog box (default is the English language). You can use FRENCH, ENGLISH, GERMAN or 'NONE' language. If you choose the 'NONE' language mode, the replace dialog box won' t display any text, so you need to provide your own text thanks to the ZN_REPLACE notification message and the UnzipSetReplaceText function.

<iLanguage> is one of the following:

Name	*Value*	*Means*
LANGUAGE_FRENCH	0	French language mode
LANGUAGE_ENGLISH	1	US language (default)
LANGUAGE_GERMAN	2	German language.
LANGUAGE_NONE	3	'None' mode.

Example

```
WinMain( ... )
   {
   UnzipInit( "TEST", "");
   UnzipSetMsgBoxTitle("Overwrite?");
   UnzipSetLanguage( LANGUAGE_FRENCH );
   ...
```

Required DLL

UZDLL20

See also...

UnzipSetMsgBoxTitle
UnzipSetReplaceText
UnzipSetReceivingWindow
UnzipSetBackgroundMode
UnzipInit

UnzipSetReceivingWindow

BOOL WINAPI UnzipSetReceivingWindow(HWND hWnd);

Description

You must use this function to specify the notification messages receiver Window (generally your main Window). These messages are sent by ExtractZipFiles and ViewFileFromZip. To cancel the notification give a NULL value for <hWnd> else give a valid HWND value for <hWnd>. This function returns TRUE if the <hWnd> value is valid (NULL or a real <hWnd>). To be sure the background mode will be turned off, give a NULL to these function before you call another UNZIP function.

Notification messages are:

Name	*Value*	*Means*	*WParam*	*LParam*
ZN_OPENFILE	WM_USER + 38	Open a file.	TRUE if OK.	Current filename
ZN_EXPANDING	WM_USER + 39	Unzipping a file.	Current fill rate.(0..100).	Current filename
ZN_CLOSEFILE	WM_USER + 40	Closing a file.	TRUE if OK.	Current filename
ZN_REPLACE	WM_USER + 53	Replace dialog is displayed.	TRUE if the ZIP version is newer.	Current filename

Example

```
char FirstFileInZip[180];
if ( CountFileInZip("MYZIP.ZIP") > 0 )
   {
   lstrcpy( FirstFileInZip, GetFileNameFromZip("MYZIP.ZIP",1));
   UnzipSetReceivingWindow( hWnd );
   ViewFileFromZip( "MYZIP.ZIP", FirstFileInZip, FALSE );
   ...
```

Required DLL

UZDLL20

See also...

UnzipSetMsgBoxTitle
UnzipSetReplaceTextUnzipSetLanguage
UnzipSetBackgroundMode
UnzipInit

UnzipSetBackgroundMode

void WINAPI UnzipSetBackgroundMode(BOOL bBackGroundMode);

Description

If you call this function with <bBackGroundMode> set to TRUE, when you will an UNZIP function, you will be free to do something else with your computer during the UNZIP process. By default, <bBackGroundMode> is set to TRUE. Setting this value to FALSE will shorten the processing required time.

Warning!

When you use the background processing mode, don' t forget that the user can also close your application before the UNZIP process is complete. That' s why you must prevent the user from closing your application during the process.

Example

```
WinMain( ... )
   {
   UnzipInit( "TEST", "");
   UnzipSetLanguage( LANGUAGE_FRENCH );
   UnzipSetBackgroundMode( FALSE );
   ...
```

Required DLL

UZDLL20

See also...

UnzipSetMsgBoxTitle
UnzipSetLanguage
UnzipSetReceivingWindow
UnzipInit

UnzipSetReplaceText

BOOL WINAPI UnzipSetReplaceText(LPCSTR szDialogTitle, LPCSTR szDialogText, LPCSTR szYesBtn, LPCSTR szNoBtn, LPCSTR szAlwaysBtn, LPCSTR szNeverBtn);

Description

This function will change the 'Replace' dialog box labels value. So you will use this function with the ZN_REPLACE notification message. If you wish to use a non standard language you will call this function with the informations the ZN_REPLACE will give to you and when you receive this message. As this function doesn' t perform any action if the dialog box is not here, in case you use Zip functions too, you will call the ZipSetReplaceText at the same time. <szDialogTitle> is the dialog box title, <szDialogText> is the replace main text ('the file ... is newer ...'), <szYesBtn> is the '&Yes' button text, <szNoBtn> is the '&No' button text, <szAlwaysBtn> is the '&Always' button text and <szNeverBtn> is the 'Ne&ver' button text. This function returns FALSE is the unzip replace dialog box is not available, TRUE in the other case.

Example

```
.../...
case ZN_REPLACE:
   {
   char szText[400];
   lstrcpy( szText, "The file  ");
   lstrcat( szText, (LPCSTR)lp ); // the file name
   lstrcat( szText, "  already exist...\n");
   if ( wp )
      lstrcat( szText, "The ZIP version is newer!");
   else
      lstrcat( szText, "The ZIP version is older.");
   // now call the both functions...
   ZipSetReplaceText( "Custom",szText,"&Yes","&No","&Always","Ne&ver);
   UnzipSetReplaceText( "Custom",szText,"&Yes","&No","&Always","Ne&ver);
   break;
   }
```

Required DLLs

UZDLL20

See also...

ExtractZipFile
UnzipSetMsgBoxTitle
UnzipSetLanguage
UnzipSetReceivingWindow
UnzipInit

Note for VENDORS!

- Deleted -
See the READ.ME file instead.

ZDEMO program

Sorry, this program is no more available with Zip Studio 2.0: Instead, the Zip Studio shell MFC source is available.

See also...

- Zip Studio Samples
- Zip Studio Shell

Terms

Description

Used terms are C and SDK language type. If you do not program with these languages, perhaps you will have some difficulties to understand them. These are some explanations:

WINAPI	FAR PASCAL (just ignore this)
int, Short	Integer value.
long	Long integer value.
UINT	Signed long integer value.
BOOL, Bool	Integer value, 0 for FALSE, anything else for TRUE.
char	Character.
char far*	String.
LPCSTR	Constant String
LPSTR, String	String.
HWND	Window' s Handle(Integer value).
HINSTANCE	Application' s Handle(Integer value).
void	Null type (empty).
COLORREF	Long integer value (Colour).
TRUE	An integer value other than 0 (1, -1 with VB).
FALSE	0

Package

<Deleted>

Registration...

Versions

The evaluation version zips and unzips into the WINDOWS\TEST directory. Also, to test this API it is a good idea to put your files in this directory. You are not allowed to distribute programs which use the evaluation version. The registered version removes the evaluation version limitations, you can distribute your works without any royalties and you will get a single licence to use the Zip Studio Shell.

Upgrade

If you registered Zip Studio 1.x you can buy the Zip Studio 2.0 upgrade for just $20. To do this, use the order form or Compuserve SWREG Form (this upgrade ID is 3833).

What product do you need?

If you register the Zip Studio 2.0 API you don' t have to register the Zip Studio Shell since the tool kit includes a licence to use this shell. If you don' t want to build some programs using the Zip Studio API you can register the Zip Studio Shell alone. We propose a single licence registration for $25, a 10 licences pack for $110 and a 100 licences pack for 380. If you wish to build your own shell (or to customise this shell) for private use you can get the Zip Studio Shell MFC (2.5) source code for $50. Please notice that this source code requires the Zip Studio 2.0 API, VBX Studio 1.2 and String Studio 2.0. This source code does not include the Zip Studiio 2.0 API source code.

How to register?

There are 3 ways to register: Of course you can order by mail (we accept US dollars, French Francs, Deutsch Mark and GB Pounds). You must order via mail when you register the 100 licences pack or when you buy a tax free product. You can also use the Compuserve SWREG Forum: To do this you must go to the SWREG Forum, then you give an ID (see below) and your credit card number. As a conclusion we accept PsL orders: PsL can collect credit cards. To register the Zip Studio 2.0 API whith PsL, use the ID #11365. Regarding the other products ask PsL for the IDs.

You can order with MC, Visa, or American Express from the Public (software) Library (PsL) by calling 1-800-2424-PSL or FAXing 1-713-524-6398. These numbers are for orders only. For questions about credit card orders, call PsL at 713-524-6394. You can also mail credit card orders to PsL at P.O.Box 35705, Houston, TX 77235-5705.

How to save $35?

We still propose a special offer pack with Zip 2.0 and Setup Studio: Setup Studio is a powerful toolkit to build nice setup programs (like the Zip Studio 2.0 installation program) and is available for $50. When you register the Zip Studio 2.0 API, you can register it with Setup Studio: Because this package costs $85 (just 15$ more), you will save $35. To register, use mail or Compuserve SWREG (ID is 2667).

Prices and SWREG IDs?

For the Zip Studio 2.0 API (with a Zip Studio Shell licence):

SWREG ID: 2456 (or 2667)

Currency:	ZIP STUDIO	ZIP STUDIO + SETUP STUDIO
Francs français:	390-	450-
US Dollars:	70-	85-
Deutsch mark:	115-	145-
GB Pounds:	46-	60-

These prices include shipping and VAT.

For the Zip Studio 2.0 API upgrade (from 1.x):

SWREG ID: 3833

Currency:	
Francs français:	130-
US Dollars:	20-
Deutsch mark:	35-
GB Pounds:	15-

Prices include shipping and VAT

For the Zip Studio Shell MFC source code:

SWREG ID: 3835

Currency:	
Francs français:	300-
US Dollars:	50-
Deutsch mark:	88-
GB Pounds:	36-

Prices include shipping and VAT

For the Zip Studio Shell (single user):

SWREG ID: 3832

Currency:	
Francs français:	150-
US Dollars:	25-
Deutsch mark:	44-
GB Pounds:	18-

Prices include shipping and VAT

For the Zip Studio Shell (licences pack):

SWREG ID: 3834
NORMAL PRICES

Currency:	10 licences	100 licences
Francs français:	690-	2300-
US Dollars:	110-	380-
Deutsch mark:	200-	660-
GB Pounds:	82-	270-

```
* These prices include shipping and VAT.        *
************************************************
```

EUROPEAN COMPANIES PRICES (*)
The following prices don' t include the French VAT: You can buy the products at these prices if (1) you register by mail, (2) you are not a French resident, (3) you are a company and you mail us your EC VAT number, and (4) this VAT number is ok.

```
************************************************
*Currency:         * 10 licences * 100 licences *
*------------------*-------------*--------------*
* Francs français: *       635-  *     1995-    *
* US Dollars:      *       103-  *      330-    *
* Deutsch mark:     *       184-  *      572-    *
* GB Pounds:       *        76-  *      234-    *
*-----------------------------------------------*
* These prices include shipping.                *
************************************************
```

And now?

Fills the order form or use SWREG or PsL registration services. In case of a mail order, send your order form with your check to HEXANET - BP 385.16 - 75768 PARIS CEDEX 16 - FRANCE. In normal cases we need one or 2 days to process your order. If you have an Internet or CIS address, we will email you your passwords. We always send a printed invoice.

Other products

- Setup Studio 2.2

If you develop under Windows 3.10 this toolkit is for you! With Setup Studio 2.2 you can build some high quality setup programs like the one you' ve used to install Zip Studio 2.0. Also, when you register Zip Studio 2.0 you can save $35- when you buy Setup Studio too (it just costs $85- and the Compuserve SWREG id for this special offer is 2667).

- VBX Studio 1.2

The ultimate VBX package for VB and C++: 15 + 2 general purpose VBXs like bitmapped listboxes, spins, tabbed dialogs or gauges for instance. At present VBX Studio is available for $55 (SWREG id is 3336) or $70 with String Studio 2.0 (SWREG id is 3521). A great value for the 'Visual' programmer!

- String Studio 2.0

A MFC CString extension for MFC (VC++) users. Now it includes over 110 routines and a DOS version is also provided. With the new CStr string class you can handle CTime objects, files, numbers, strings items and much more. Useful with VBX Studio. Source code is available on Compuserve for just $35- (SWREG id is 3003) or $70 with VBX Studio 1.2 (SWREG id is 3521).

See also...

Overview
Order form
Shareware

Shareware?

Zip Studio is a ShareWare:

You can try it as you want before you decide or not to register. The only one way we have to sell enough licences to support this software, is to distribute it all over the world. So, if you think this product will help one of your friends, don' t hesitate to give him a copy of this product: This is your interest!

When you distribute Zip Studio, don' t change any file and give the complete package. If you UL Zip Studio on a BBS, please name it ZIPSTD20.ZIP or ZIPSTD20.EXE. Obviously, if you register, you can' t distribute the registered version! Distributing programs made with Zip Studio unregistered version is forbidden!

Order form

To register you can print this help page and manually fill it, or use ORDER.TXT. Unreadable orders will be ignored!

--- Zip Studio 2.0 ORDER FORM ---

Name

```
COMPANY:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
NAME:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
FIRST NAME:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
ATTENTION TO:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
ADDRESS:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
ZIP:_ _ _ _ _ _ _ _ _ _
TOWN:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
COUNTRY:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
COMPUSERVE ID: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
EMAIL: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
WHERE DID YOU FIND THE ZIP STUDIO?:_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

Product(s)

```
QUANTITY:_ _ _
CURRENCY:_ _ _ _ _ _ _
EC VAT NUMBER: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
PRODUCT(S):_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

For the Zip Studio 2.0 API (with a Zip Studio Shell licence):

SWREG ID: 2456

*Currency:	* ZIP STUDIO	* ZIP STUDIO + SETUP STUDIO
* Francs français:	390-	450-
* US Dollars:	70-	85-
* Deutsch mark:	115-	145-
* GB Pounds:	46-	60-

* These prices include shipping and VAT.

For the Zip Studio Shell MFC source code:

SWREG ID: 3835

*Currency:	*
* Francs français:	300-
* US Dollars:	50-
* Deutsch mark:	88-
* GB Pounds:	36-

* Prices include shipping and VAT*

For the Zip Studio 2.0 API upgrade (from 1.x):

SWREG ID: 3833

Currency:	
Francs français:	130-
US Dollars:	20-
Deutsch mark:	35-
GB Pounds:	15-

Prices include shipping and VAT

For the Zip Studio Shell (single user):

SWREG ID: 3832

Currency:	
Francs français:	150-
US Dollars:	25-
Deutsch mark:	44-
GB Pounds:	18-

Prices include shipping and VAT

For the Zip Studio Shell (licences pack):

SWREG ID: 3834
NORMAL PRICES

Currency:	10 licences	100 licences
Francs français:	690-	2300-
US Dollars:	110-	380-
Deutsch mark:	200-	660-
GB Pounds:	82-	270-

These prices include shipping and VAT.

EUROPEAN COMPANIES PRICES (*)
The following prices don' t include the French VAT: You can buy the products at these prices if (1) you register by mail, (2) you are not a french resident, (3) you are a company and you mail us your EC VAT number, and (4) this VAT number is ok.

Currency:	10 licences	100 licences
Francs français:	635-	1995-
US Dollars:	103-	330-
Deutsch mark:	184-	572-
GB Pounds:	76-	234-

These prices include shipping.

****

Send this with your check to

HEXANET,
BP 385.16
75768 PARIS CEDEX 16
FRANCE

What' s new?

Corrected bugs...
Most of the Zip Studio routines were re-written.

Building ZIP files list...
Previous release don' t build ZIP files list, so, to build the files list yourself you were using the <GetZFile...> functions: It was too slow... Now we use <GetZList> to build a ZIP files list and this method is quite fast with big files. The function uses an hidden (but visible) system listbox. You can use many sort modes with this list. To retrieve informations regarding an item you must use the <GetZipItem> function.

Using encrypted ZIP files...
Zip Studio supports PKWARE' s PKZIP encrypted Zip files. To build an encrypted ZIP file you will give a password to <AddFileToZip> and to unzip an encrypted Zip file you will call <UnzipSetPassord>. To simplify your source code, the Zip Studio Shell can display a PASSWORD dialog box in case the password is not valid for a file: To do this, call <SetAskPassword>.

Using multiple parts ZIP files...
Now Zip Studio can split a ZIP file thanks to <ZipSplit>. The splitted parts can be used with PKUNZIP once you join them thanks to the Zip Studio or, some third party sharewares, or the DOS command COPY <file> /B... To join these files with Zip Studio you will call <ZipJoin>. These functions can use removable drives and includes customisable MessageBoxes. Zip Studio doesn' t handle PKWARE's PKZIP 2.04g multiple parts Zip files.

The new Zip engine...
Zip Studio 2.0 lets you choose the compression mode with <ZipSetMode>: The NORMAL mode is about 500-600% faster than the previous release.

Now you can cancel some actions...
To cancel an Extraction or View action you will call <UnzipCancel>. To cancel the <AddFileToZip> process you will call <ZipCancel>. The function will try to stop the process as soon as possible. Regarding <ZipCancel>, if you use this function you must reset the CANCEL flag each time you call an UZDLL20 routine.

The Zip Studio Shell...
This shell is a natural extension for the new API. Its MFC 2.5 source code replaces the old ZDEMO sample and is available for $50. You will need about 1Mb for this source code and the final project will require more than 5Mb... Also, to build this sample you will need String Studio 2.0, VBX Studio 1.2 and (of course) the Zip Studio 2.0 API. The shell doesn' t include any hard coding string: All these strings are in the resource. That' s why you can easily translate them. The Zip Studio Shell was designed for buisness: We propose very attractive licences packs prices (from $3.8 per licence). We hope the ZIP Studio 2.0 is a complete solution for your needs. See the Zip Studio Shell page for details.

A word about the VBX...
VB users will probably use the Zip Studio VBX (ZIPSTD20.VBX): This VBX supports all the new features, you don' t have to worry any more regarding the C strings management and it' s quite easier to use the VBX instead of direct DLLs calls. See the VBX page for more informations.

A word about the 32Bits version...
A 32Bits version is in studying but we will probably wait for Windows 95 or Visual Basic 4 before we will release it. If you need a 32Bits version, send us an EMail so that we can have an idea about the number of our customers who take an interest in such a version....

Upgrade

Zip Studio 1.x users can buy the Zip Studio 2.0 upgrade for just $20. To do this use the Order Form or Compuserve SWREG Forum (the ID is 3833).

The following lines concern Zip Studio 1.x:

Handling strings with VB...

VB users find many troubles with our DLL strings (specially with GetFileNameFromZIP). With this new release if you are a VB user, you will use the VBX interface to forget this problems but, if you wish to use the DLLs API, this is a way to handle the Zip Studio strings with VB:

As you know a C String is NUL terminated: the last character has a zero value. With VB, you can create this string like this:

```
szResult$ = String(255,0)
```

This will create a NULL terminated string with 255 characters at max. Now, you will call a Zip Studio function with a string buffer, like VBGetFileNameFromZIP (see later):

```
VBGetFileNameFromZIP zipfile$, fnum%, szResult$
```

If everything is Ok your szResult$ string will contain the function result and trailing zero characters. Maybe you don' t need to remove these characters (try it). To delete these call:

```
iZero% = InStr(szResult$, Chr$(0))
szResult$ = Left$( szResult$, iZero% - 1 )
```

Also Zip Studio 2.0 provides 3 new functions for the VB users (if you are a VB user and in case you use the DLLs, you must call these functions instead of the original functions):

```
      Declare Sub  VBGetFileNameFromZIP Lib "UZDLL20.DLL"( ByVal szFileName$,
ByVal iFileNumber%, ByVal szResult$ )
      Declare Sub  VBGetZFileDate Lib "UZDLL20.DLL"( ByVal
szZIPFileName$, ByVal szFileName$, ByVal szResult$ )
      Declare Sub  VBGetZFileTime Lib "UZDLL20.DLL"( ByVal
szZIPFileName$, ByVal szFileName$, ByVal szResult$ )
```

Some improvements...

Now Zip Studio has a built-in German language support: See ZipSetLanguage for more informations. Also, the library can use others languages: To use a non standard language, give LANGUAGE_NONE to the ZipSetLanguage and UnzipSetLanguage. Also, the 'Replace' dialog box will send you a notification message (ZN_REPLACE) with the filename. At this time you will use the ZipSetReplaceText and UnzipSetReplaceText new functions to provide your own labels texts. Also, I' ve added new functions: IsFileInZip, GetZFileFullInfos, GetZFileIsDir ... Now Borland Pascal 7 users can use the ZIP\UNZIP API thanks to the Brad STOWERS work. You can have a look at the Tips chapter for FAQ answers.

References...

It' s a little too sson to give you a list of softwares built with the Zip Studio API. However you will probably see these next monthes some commercial softs or sharewares which use the Zip Studio, like the Huw MILLINGTON Windows Grep 1.5x...

See also...

- Overview
- Tips
- The VBX interface

Using Zip Studio with VB...
Zip Studio Samples

VBGetFileNameFromZIP

Declare Sub VBGetFileNameFromZIP Lib "UZDLL12.DLL"(ByVal szFileName$, ByVal iFileNumber%, ByVal szResult$)

Description

Use this sub instead of GetFileNameFromZIP. <szFileName> is the ZIP file name, <iFileNumber> is the file number you wish to retrieve the name (0 indexed) and <szResult> is the result buffer.

Example

```
szResult$ = String(255,0)
VBGetFileNameFromZIP zipfile$, fnum%, szResult$
iZero% = InStr(szResult$, Chr$(0))
szResult$ = Left$( szResult$, iZero% - 1 )
```

See also...

GetFileNameFromZIP
VBGetZFileDate
VBGetZFileTime
Using Zip Studio with VB...

VBGetZFileDate

Declare Sub VBGetZFileDate Lib "UZDLL12.DLL"(ByVal szZIPFileName$, ByVal szFileName$, ByVal szResult$)

Description

Use this sub instead of GetZFileDate. <szZIPFileName> is the ZIP file name, <szFileName> is the file name of the file you wish to retrieve the date and <szResult> is the result buffer.

Example

```
szResult$ = String(25,0)
VBGetZFileDate zipfile$, "MYFILE.TXT", szResult$
iZero% = InStr(szResult$, Chr$(0))
szResult$ = Left$( szResult$, iZero% - 1 )
```

See also...

GetZFileDate
VBGetFileNameFromZIP
VBGetZFileTime
Using Zip Studio with VB...

VBGetZFileTime

Declare Sub VBGetZFileTime Lib "UZDLL20.DLL"(ByVal szZIPFileName$, ByVal szFileName$, ByVal szResult$)

Description

Use this sub instead of GetZFileTime. <szZIPFileName> is the ZIP file name, <szFileName> is the file name of the file you wish to retrieve the time and <szResult> is the result buffer.

Example

```
szResult$ = String(25,0)
VBGetZFileTime zipfile$, "MYFILE.TXT", szResult$
iZero% = InStr(szResult$, Chr$(0))
szResult$ = Left$( szResult$, iZero% - 1 )
```

See also...

GetZFileTime
VBGetFileNameFromZIP
VBGetZFileDate
Using Zip Studio with VB...

Using Zip Studio with VB

Overview

VB users find many troubles with the DLLs strings (specially with GetFileNameFromZIP). With this new release if you are a VB user, you will use the VBX interface to forget this problems but, if you wish to use the DLLs API, this is a way to handle the Zip Studio strings with VB:

Using C Strings

As you know a C String is NUL terminated: the last character has a zero value. With VB, you can create this string like this:

```
szResult$ = String(255,0)
```

This will create a NULL terminated string with 255 characters at max. Now, you will call a Zip Studio function with a string buffer, like VBGetFileNameFromZIP (see later):

```
VBGetFileNameFromZIP zipfile$, fnum%, szResult$
```

If everything is Ok your szResult$ string will contain the function result and trailing zero characters. Maybe you don' t need to remove these characters (try it). To delete these call:

```
iZero% = InStr(szResult$, Chr$(0))
szResult$ = Left$( szResult$, iZero% - 1 )
```

Specific functions

Also Zip Studio 2.0 provides 3 new functions for the VB users (if you are a VB user and in case you use the DLLs, you must call these functions instead of the original functions):

```
      Declare Sub  VBGetFileNameFromZIP Lib "UZDLL20.DLL"( ByVal szFileName$,
ByVal iFileNumber%, ByVal szResult$ )
      Declare Sub  VBGetZFileDate Lib "UZDLL20.DLL"( ByVal
szZIPFileName$, ByVal szFileName$, ByVal szResult$ )
      Declare Sub  VBGetZFileTime Lib "UZDLL20.DLL"( ByVal
szZIPFileName$, ByVal szFileName$, ByVal szResult$ )
```

See also...

- VBGetFileNameFromZIP
- VBGetZFileDate
- VBGetZFileTime
- Zip Studio Samples

Tips

- Using the Zip functions (DLLs)...
- Using the Unzip functions (DLLs)...
- Using Zip Studio (DLLs) with VB...
- Using the VBX...
- How to build a ZIP files list?
- How to Cancel an action?
- Zipping to a network...
- Using encrypted files...
- Using multiple parts ZIP file...

See also...

- Zip Studio Samples

The Zip Studio VBX

Zip Studio 2.0 provides a new VBX interface for all the DLLs API. You can use this VBX with or without the Zip Studio functions but, you when you use the VBX, you must distribute it with all the DLLs with your applications. If you choose to register you will nee to use the VBXREG2 program to create the licence file.

Overview (VBX)

Advantages/Disadvantages

This new Zip Studio release lets you use the Zip Studio API (thanks to DLLs calls) and/or a VBX control. Using the VBX is far much simple than using the API: You just fill some labels at design time in your 'Visual' development system and most properties perform the necessary operations for Zip and Unzip operations. Also VB users will be pleased to receive the Zip Studio notifications messages and to use C strings without additional handling. So what' s the DLLs interest? First of all with the VBX you must distribute the full Zip Studio DLLs whereas with the API you can use only UZDLL20.DLL and its unzip functions for instance. Also using the VBX requires more user's hard disk space and it is not as fast as the direct API calls.

How to install the VBX?

Of course this depends on your Compiler/Development system. In most case you will put all the Zip Studio DLLs and the ZIPSTD20.VBX in the WINDOWS\SYSTEM directory. Also if you plan to use the <Help> property, put thie ZIP.HLP help file into this directory too. Now call your Compiler and choose 'Add file', 'Install controls' or something like this and choose the ZIPSTD20.VBX. You will copy or include ZIPVBX.H or ZIP.BAS to your project to make the job easier. That' s all. To redistribute your application do not forget to give ZIPSTD20.VBX! Our other toolkit, Setup Studio 2.2 can help you to design a setup program to make it perfectly and you can save 35 dollars if ou register the both at the same time.

How to use the VBX?

If you don' t know anything about the Zip Studio API, have a look at this now! Yes you must have an idea about the Zip Studio treatments before you start to use the VBX... Now, open your project (or resources editor) and click on the 'ZIP' palette button, make a drag' n drop to your form (or dialog box) and you will see a (nice) little 'ZIP' icon! Have a look at this object 'properties', you will notice that not all the properties are 'visible'. The properties you can watch are the common parameters you maybe don' t need to change at run-time. Also you can change the <ZipName>, <FileNameOrMask> or <UnzipDestDir> at design time to make your application testing more easy to do. As you can guess the most important parameter is the <ZipName>, so you will set this parameter before any other property. Also, <FileNameOrMask> is used for Zip and Unzip functions, so its mean depends on the function you will call, that' s why it can be a file name, a mask (like '*.*') or a set of masks (like '*.BAS *.FRM') for Unzip functions. Of course you will use just 1 Zip Studio VBX in your project!

Using the indexed properties...

The Zip Studio VBX contains 8 indexed properties (beginning with a 'Z'). These properties are used to retrieve informations regarding a file in the ZIP file. For instance, if you wish to know the last modification date of a file in your current Zip file, you will read 'VBZip1.ZFileDate(0)' that' s all. However, if you wish to retrieve all the informations at the same time, you will use the <ZFileFullInfos> indexed property.

Other VBXs?

VBX Studio is a package with more than 15 general purpose VBXs to build state of the art dialog box. Also VBX Studio was designed to save your development times as VBTabFList do. VBX Studio 1.2 costs no more than 55$-, the Zip file is available on the Compuserve MSBASIC Forum and you can register it on Compuserve SWREG (id is 3336 or 3337 with Setup Studio).

See also...

- Properties
- Events
- VBXREG registration utility
- The VB sample

VBX Properties

The ZIP Studio 2.0 VBX performs direct calls to the Zip Studio API. That' s why you will find a complete help with the API functions descriptions ('Related function(s)').

Common properties

Action [2.0]

Type: Short
Related function(s): AddFileToZip , ZipDeleteFiles , ZipRepare , ExtractZipFiles
This is the main special property. Thanks to this property you will choose an action (an operation) like ADDING a file to the ZIP file (1), DELETING a file from the ZIP file (2), EXTRACTING the ZIP file contents (3), VIEWING a file (4), FIXING the ZIP file (5), DELETING the ZIP comment (6), .BUILD THE ZIP LIST (7), SORTING THE ZIP LIST (8), CANCELING AN ACTION (9), JOINING A FILE (10), SPLITTING A FILE (11), or TESTING THE ZIP FILE (12). Of course you must set all the necessary properties before you call this one (specially the <ZipName>, <FileNameOrMask> and <UnzipDestDir>). When the operation is done an <ActionResult> notification message will be sent out but you can also read the <ActionResult> property. Regarding the new actions SPLIT and JOIN, you must fill the <Split...> and <Join...> properties and these action don' t use <ActionResult> but <JoinResult> and <SplitResult>. Also, when you build the ZIP list, the result will be available in the <ZListResult> property.

ActionResult

Type: Short
Related function(s): See <Action>.
This property contains the last operation result. Of course this value depends on the last function you' ve called, then you must have a look at the corresponding function in this help file to know the result codes.

BackgroundProcessing

Type: Bool
Related function(s): UnzipSetBackgroundMode
Set this property to TRUE (-1) if you wish the Zip and Unzip operations run in background. If you set this property to FALSE you won' t receive the VBX notification messages.

Comment

Type: String
Related function(s): ZipSetComment , GetZipCommentLength , GetZipComment
This property automatically performs an action when you use it. So, when you read it, the ZIP comment will be extracted and when you change this property, the ZIP comment will be changed or added and an <ActionResult> notification message will be sent out. Notice than this property can' t delete the comment: To do this you will call the <Action> property instead.

FileNameOrMask

Type: String
Related function(s): AddFileToZip , ZipDeleteFiles , ExtractZipFiles
This is the name of the file you want to Zip or Unzip. It can be a mask or a set of masks depending on the action you will perform. Zip functions doesn' t allow lists (of files or filter) whereas Unzip functions can use such lists.

Help

This special property calls this help file if this one is in the same directory as the VBXSTD13.VBX!

Language

Type: Short
Related function(s): UnzipSetLanguage , ZipSetLanguage
Specify the language to use with the 'Replace' dialog box. If you wish to provide a customised 'Replace' dialog box, set this property to NONE and call the ZipSetReplaceText and UnzipSetReplaceText when you receive the <Replace> notification message. Also this new Zip Studio version supports the German language too.

OverwiteMode

Type: Short
Related function(s): AddFileToZipExtractZipFiles

You will choose the Overwrite mode thanks to this property and the 'UPDATE' mode is not available for UNZIP operations. The <ZipReplaceFlag> and <UnzipReplaceFlag> can be used with this property if you' ve given more than one mask to the <FileNameOrMask> for instance. Please notice the overwrite constants for the VBX are not the same than the API and when you use the VBX, these constants are the same for Zip and Unzip operations.

szPassword [2.0]
Type: String
Related function(s): AddFileToZipExtractZipFiles , UnzipSetPassword
This is the password you will use to build encrypted files or to unzip encrypted files. To disable this encryptation, give "".

vbxAbout...
Displays the VBX 'about' box.

ZipName
Type: String
Related function(s): - none -
This is the name of your main ZIP file. Also this file can exist or not. At run time if you give '?' to this property the browse dialog box will appear.

Zip properties

CompressionMode [2.0]
Type: Short
This property will set the compression mode (method) for zipping. Default is NORMAL.

Recurse
Type: Bool
Related function(s): AddFileToZip
Set this property to TRUE (-1) if you wish to recurse the sub directories when you set the <Action> property to ZIP.

StorePath
Type: Bool
Related function(s): AddFileToZip
Set this property to TRUE (-1) if you wish to store your files paths in the ZIP file.

ZipReplaceFlag
Type: Short
Related function(s): ZipSetLanguage , ZipSetReplaceText , ZipGetReplaceFlag ,
If the user choose a different replacement mode when the 'Replace' dialog box is displayed, sometimes you need to retrieve the user choice replacement mode. This function also returns the last replacement mode for Zip operations. If you use this property, you will give its value to the <OverwriteMode> property.

Unzip and information properties

AskForPassword [2.0]
Type: Bool
Related function(s): SetAskPassword
Set this property to TRUE if you wish the Zip Studio asks the user for a password if the password is not valid for a file.

FileInfos
Type: String
Related function(s): GetZFileFullInfos
This property will give you the <GetFileInfos> result. So, to read this property, you must give a correct value to the <GetFileInfos> property before. The string contains the file name, the original file size, the compressed size, the date,

the time and the used compression method. So the string is like this: MYFILE.TXT;125;87;01/01/94;08:15pm;8.

GetCount
Type: Short
Related function(s): CountFileInZip
Returns the number of the files (and dirs) in the current ZIP file.

GetFileInfos
Type: String
Related function(s): GetZFileFullInfos
When you need to retrieve all the informations for a file in your ZIP file use this property. You must give the file name to this property and you will read the result in <FileInfos>.

IsDir
Type: Bool
Related function(s): GetZFileIsDir
This property will be set to TRUE (-1) if the <FileNameOrMask> specifies a directory (and not a true file) in the <ZipName> ZIP file.

IsFileInZip
Type: Bool
Related function(s): IsFileInZip
Returns TRUE (-1) if the <FileNameOrMask> file specification is in the ZIP file.

IsFileUnzipable
Type: Bool
Related function(s): IsFileUnzipable
Returns TRUE (-1) if the <FileNameOrMask> file seems to be a valid DOS file name. In most cases you don' t need to call this property because ZIP Studio translates the UNIX files name into DOS files name. This property will returns FALSE if you give a path with your file.

IsItAZipFile
Type: Bool
Related function(s): IsThisFileAZipFile
Returns TRUE (-1) if the <ZipName> file is really a ZIP file!

RecreateDir
Type: Bool
Related function(s): ExtractZipFiles
Set this property to TRUE if you wish to recreate the paths specified in your ZIP file (if any).

UnzipDestDir
Type: String
Related function(s): ExtractZipFiles
This property is not available for the unregistered users (in this case all files will be unzipped into the WINDOWS\TEST directory). This is the name of the directory in which you wish to unzip your files. This name can end with a '\' or not. At run time, if you give a '?' to this property, the browse dialog box will be displayed.

UnzipReplaceFlag
Type: Short
Related function(s): ExtractZipFiles , GetQueryFlag ,
If the user choose a different replacement mode when the 'Replace' dialog box is displayed, sometimes you need to retrieve the user choice replacement mode. This function also returns the last replacement mode for Unzip operations. If you use this property, you will give its value to the <OverwriteMode> property.

ViewAsText
Type: Bool
Related function(s): ViewFileFromZip ,
Set this property to TRUE if you want to display all the ZIP files as text files (using the notepad for instance) instead of viewing them with their Winfile associated editor.

Join properties

JoinDestDir [2.0]
Type: String
Related function(s): ZipJoin
Give the destination directory for the ZIP file. If you give '?' a dialog box will be displayed.

JoinMsg [2.0]
Type: String
Related function(s): ZipJoin
This is the Join message box tiltle. %1 will be replaced with the diskette number and %2 with the number of diskettes.

JoinResult [2.0]
Type: Short
Related function(s): ZipJoin
The last join operation result.

JoinSrcFile [2.0]
Type: String
Related function(s): ZipJoin
This is the name of a part file (*.?_?). If you give '?' a dialog box will be displayed.

JoinTitle [2.0]
Type: String
Related function(s): ZipJoin
This is the Join message box tiltle. %1 will be replaced with the diskette number and %2 with the number of diskettes.

Split properties

SplitDestDir [2.0]
Type: String
Related function(s): ZipSplit
Give the destination directory for the part files. If you give '?' a dialog box will be displayed.

SplitDiskettePause [2.0]
Type: BOOL
Related function(s): ZipSplit
If set to TRUE the Zip Studio Shell will wait for a new diskette.

SplitFirstBlockSize [2.0]
Type: Long
Related function(s): ZipSplit
This is the size (in bytes) of the first part file.

SplitMsg [2.0]
Type: String
Related function(s): ZipSplit
This is the Split message box tiltle. %1 will be replaced with the diskette number and %2 with the number of diskettes.

SplitNextBlockSize [2.0]
Type: Long
Related function(s): ZipSplit
This is the size (in bytes) of the next parts files.

SplitResult [2.0]
Type: Short

Related function(s): ZipSplit
The last split operation result.

SplitSrcFile [2.0]
Type: String
Related function(s): ZipSplit
This is the name of an existing ZIP file . If you give '?' a dialog box will be displayed.

SplitTitle [2.0]
Type: String
Related function(s): ZipSplit
This is the Split message box tiltle. %1 will be replaced with the diskette number and %2 with the number of diskettes.

ZList properties

ZListDay [2.0]
Type: Short
Related function(s): GetZList , GetZipItem
The last modification date (day) of the ZList current item (selected thanks to the <ZListIndex> property).

ZListDir [2.0]
Type: String
Related function(s): GetZList , GetZipItem
The relative directory of the ZList current item (selected thanks to the <ZListIndex> property).

ZListIndex [2.0]
Type: Short
Related function(s): GetZList , GetZipItem
Before you can read the following <ZList..> properties, you must go to a specified item in the list (0 indexed). We used this method for speed because we read the Zip Item just one time, then we read the <ZList...> property.

ZListIsDir [2.0]
Type: BOOL
Related function(s): GetZList , GetZipItem
TRUE if the ZList current item (selected thanks to the <ZListIndex> property) is a directory.

ZListIsEncrypted [2.0]
Type: BOOL
Related function(s): GetZList , GetZipItem
TRUE if the ZList current item (selected thanks to the <ZListIndex> property) is encrypted.

ZListFrom [2.0]
Type: Short
Related function(s): GetZList
If you use a too big ZIP file, maybe the ZList doesn' t contain all the files informations. In this case the <ZListResult> is the last read file. So, you can give this index to this property to retrieve the next files infos (to build a new ZList beginning with the item <ZListFrom>).

ZListHour [2.0]
Type: Short
Related function(s): GetZList , GetZipItem
The last modification time (hour) of the ZList current item (selected thanks to the <ZListIndex> property).

ZListMethod [2.0]
Type: Short
Related function(s): GetZList , GetZipItem GetZCompressMethod
The compression method of the ZList current item (selected thanks to the <ZListIndex> property).

ZListMin [2.0]
Type: Short
Related function(s): GetZList , GetZipItem
The last modification time (minute) of the ZList current item (selected thanks to the <ZListIndex> property).

ZListMonth [2.0]
Type: Short
Related function(s): GetZList , GetZipItem
The last modification date (month) of the ZList current item (selected thanks to the <ZListIndex> property).

ZListName [2.0]
Type: String
Related function(s): GetZList , GetZipItem
The name of the ZList current item (selected thanks to the <ZListIndex> property). This name doesn' t include any path and can be a blank string if the ZList item is a directory.

ZListOriginalSize [2.0]
Type: Long
Related function(s): GetZList , GetZipItem
The original (not compressed) size in bytes of the ZList current item (selected thanks to the <ZListIndex> property). To retrieve the compressed size, you will use this value with the <ZListRate> property.

ZListRate [2.0]
Type: Short
Related function(s): GetZList , GetZipItem
The compressio rate of the ZList current item (selected thanks to the <ZListIndex> property).

ZListResult [2.0]
Type: Short
Related function(s): GetZList
There are 3 cases: The ZIP file can' t be read, the property value is -1. The Zip file was successfully loaded and all items were read, the property value is 0. The ZIP file is too big, this property value is the last read file. So, you can give this index to the <ZListFrom> property to retrieve the next files infos (to build a new ZList beginning with the item <ZListFrom>).

ZListYear [2.0]
Type: Short
Related function(s): GetZList , GetZipItem
The last modification date (year (00-99)) of the ZList current item (selected thanks to the <ZListIndex> property).

ZListZortMode [2.0]
Type: Short
Related function(s): GetZList, UnzipSortZList
You must use this property to set the Zip List sort mode. To make the change call the <Action> property.

ZFile properties

ZFileCompSize
Type: Long (array)
Related function(s): GetZFileCompressedSize ,
The compressed length of a file X in a ZIP file (0 indexed), in bytes.

ZFileDate
Type: String (array)
Related function(s): GetZFileDate ,
The last modification date of a file X in a ZIP file (0 indexed). This value depends on the user Windows configuration.

ZFileFullInfos
Type: String (array)
Related function(s): GetZFileFullInfos
This property is the same as <FileInfos> but it works with a file X in a ZIP file (0 indexed). Also, you don' t have to give the filename to use it, you will give an index instead.

ZFileIsDir
Type: Bool (array)
Related function(s): GetZFileIsDir
This property is the same as <IsDir> but it works with a file X in a ZIP file (0 indexed). Also, you don' t have to fill the <FileNameOrMask> property to use it, you will give an index instead.

ZFileIsEncrypted [2.0]
Type: Bool (array)
Related function(s): GetZFileIsEncryptedr
Set to TRUE if the file X in a ZIP file (0 indexed) is encrypted.

ZFileMethod
Type: Short (array)
Related function(s): GetZCompressMethod ,
The compression method of a file X in a ZIP file (0 indexed). See for a list of possible values.

ZFileName
Type: String (array)
Related function(s): GetFileNameFromZIP ,
The name of a file X in a ZIP file (0 indexed).

ZFileOriginalSize
Type: Long (array)
Related function(s): GetZFileOriginalSize ,
The original (before the compression) length of a file X in a ZIP file (0 indexed), in bytes.

ZFileTime
Type: String (array)
Related function(s): GetZFileTime ,
The last modification time of a file X in a ZIP file (0 indexed).

ZFileUnzipable
Type: Bool (array)
Related function(s): IsFileUnzipable
Set to TRUE (-1) if the file X in a ZIP file (0 indexed) seems to have a DOS file name.

See also...

- VBX Overview
- Events
- VBXREG registration utility
- The VB sample

VBX Events

To receive the Zip Studio VBX events (notification messages), you must set the <BackgroundProcessing> property to TRUE.

Compute

Params: NumberOfFiles as integer
Related function(s): AddFileToZip
Before a Zip operation (like AddFileToZip) you will receive this notification message because in case the Zip file contains more than 500 files, the process will require a lot of time!

UnzipOpenFile

Params: FileOk as integer, FileName as string
Related function(s): ExtractZipFiles
This message will be sent each time a file (<FileName>) is open during an Unzip operation and <FileOk> will be set to FALSE if an error occurs during this process.

UnzipCloseFile

Params: FileOk as integer, FileName as string
Related function(s): ExtractZipFiles
This message will be sent each time a file (<FileName>) is closed during an Unzip operation and <FileOk> will be set to FALSE if an error occurs during this process.

Deleting

Params: FileName as string
Related function(s): AddFileToZip
This message will be sent each time a file <FileName> is deleted during a Zip operation.

Zipping

Params: CompSize as integer, FileName as string
Related function(s): AddFileToZip
This message is useful to retrieve the compressed size of a file (<FileName>) in the current ZIP file during the Zip process. <CompSize> is the current file size in KBytes when this message is sent out.

FileZipped

Params: CompRate as integer, FileName as string
Related function(s): AddFileToZip
The same as the Zipping message but this message is sent when the Zip process is done and <CompRate> is the final compression rate.

Writing

Params: ZipName as string
Related function(s): AddFileToZip
Now the Zip file is complete, we replace the old Zip file (if any) with the new one. <ZipName> is the name of the main ZIP file.

Replace

Params: IsNewer as integer, FileName as string
Related function(s): ExtractZipFiles , AddFileToZip
This messages is useful if you wish to provide your own 'Replace' dialog labels. This message is sent out when this dialog box is displayed on the screen, then you will use to change the labels thanks to this event informations: <IsNewer> is set to TRUE is the ZIP file version is newer, and, <FileName> is the name of the file to overwrite (or not).

ActionResult

Params: Action as integer, Result as integer
Related function(s): - All 'actions' functions -
Probably the most important notification message. This message will be send even if you' ve set the <BackgroundProcessing> to FALSE. This message will be sent out each time you perform an action (like adding a file in a ZIP or deleting the ZIP comment). <Action> is the action id (see the <Action> property for the list) or 7 in

case you' ve set the comment and <Result> is the action result. <Result> depends on the action you' ve performed, so you must have a look at the corresponding 'related function' to know it. Also, the <ActionResult> property will be the same as the <Result> parameter.

CommentTooLong
Params: CommentLength as long
Related function(s): -
This message will be sent out in case the comment exceeds 16 Kb, <CommentLength> is the length in bytes of that guilty ZIP comment.

Expanding
Params: iRate as integer, FileName as string
Related function(s): ExtractZipFiles
Sent out during the unzipping processes, <iRate> is the rate and <FileName> is the current file name.

Encrypt [2.0]
Params: iRate as integer, FileName as String
Related function(s): AddFileToZip
Sent out when the Zip Studio encrypt a file. <iRate> is the rate and <FileName> is the current file name.

JoinDoneName [2.0]
Params: ZipName as String
Related function(s): ZipJoin
The new ZIP file is ready to be used. <FileSize> is the new ZIP file name.

JoinDoneSize [2.0]
Params: FileSize as Long
Related function(s): ZipJoin
The new ZIP file is ready to be used. <FileSize> is the new ZIP file length.

Joined [2.0]
Params: PartNumber as integer, FileSize as Long
Related function(s): ZipJoin
Sent out when a part is joinded. <PartNumber> is the piece file number, <FileSize> is the new ZIP file length.

Joining [2.0]
Params: iRate as integer, PieceName as String
Related function(s): ZipJoin
Sent out when the Zip Studio is joining a ZIP file. <iRate> is the rate (it can be wrong if the parts have different sizes) and <PieceName> is the current file name.

NetRead, NetWrite [2.0]
Params: iRate as integer, FileName as String
Related function(s): AddFileToZip
Sent out when the Zip Studio uses a temporary file to zip on a network. <iRate> is the rate and <FileName> is the current file name.

NewPassord [2.0]
Params: FileName as String, Password as String
Related function(s): ExtractZipFiles , SetAskPassword
Sent out when a new password is required.

Splitted [2.0]
Params: PartNumber as integer, FileSize as Long
Related function(s): ZipSplit
Sent out when a part is done. <PartNumber> is the piece file number, <FileSize> is the new part file length.

Splitting [2.0]
Params: Size as integer, PieceName as String
Related function(s): ZipSplit
Zip Studio is splitting the Zip file. <Size> is the part size in KBytes, PieceName is the part name.

Test [2.0]

Params: bOk as integer, FileName as String
Related function(s): IsThisFileAZIPFile
Sent out when the Zip Studio tests a ZIP file. <bOk> is TRUE if the file is OK and <FileName> is the current file name.

See also...

- VBX Overview
- Properties
- VBXREG registration utility
- The VB sample

VBXREG2 registration utility

Please notice you must use VBXREG2.EXE and not VBXREG.EXE.

Basic considerations

As you know the API requires 2 passwords you receive when you register. This passwords will be used with the initialization functions . To use the VBX registered version the process is different because there is no functions! So, the Zip Studio VBX search for a 'licence' file on your system when you use the control at design time. VBXREG is a small program which uses your passwords to build this licence file.

Using VBXREG2

Once you are a registered user, call this program and then give your passwords as specified. If the licence file can be create you will see a message box, otherwise your passwords are incorrects. Now, you will reopen all your dialog boxes (or forms) to make the change. That' s all. Of course you must not distribute the licence file (the LIC file)!!! If you are a registered user and in case the warning still appear, use the VBXREG again.

See also...

- VBX Overview
- The VB sample

The VB sample

Overview

Now Zip Studio 2.0 is provided with a VB 2.0 sample source code. This sample uses the Zip Studio VBX (not the DLL API) and it has no interest outside the Zip Studio demonstration: this program is very simple and the forms aren' t well designed!

Using this sample

To use this sample you must have VB 2.0! Load the project (open project [ZSVBDEMO.MAK]) and then press the [F5] key. First of all give a ZIP file name using the first <Browse...> button and fill the <Unzip dest dir> and the <FileNameOrMask> edit controls. Now you can set some options or use the main commands. These commands will use the values you' ve given to the edit controls (and not the names in the listbox). Now you can double click on an item in the listbox.

Remark

Notice that the VB Sample use the VBX Studio 1.2, VBFLIST.VBX. The control build bitmapped files.

See also...

- VBX Overview
- Zip Studio Samples
- Using Zip Studio with VB

The BP7 Demo and Interface

Overview

Now Borland Pascal 7 users can use the ZIP\UNZIP API thanks to the Brad STOWERS work. This sample contains an EXE with source code (like VC++ ZDEMO) and the interface for the PASCAL language. This program and source code is given as it (see the pascal files for Brad STOWERS terms).

Warning!

Maybe you will have to add some notification messages declarations.

See also...

Zip Studio Samples

Sample programs

We' ve tried to make our Zip API easy to use with the most of compiler/interpreter, that' s why we give you 2 sample programs:

1> The ZPASDEMO is a BP7 sample and the BP7 interface for the API. See BP7DEMO.
2> The VB sample (ZIPTEST.MAK) requires VB 2.0 and uses the new Zip Studio VBX. See VB Sample.

Also notice that both the VB Sample use the VBX Studio 1.2, VBFLIST.VBX. This control build bitmapped files lists.

The ZDEMO MFC source code doesn' t exist any more. The Zip Studio Shell replaces this old program. If you would like to study the Zip Shell source code or to customise this tool (for your own use), you can register the Zip Studio Shell MFC 2.5 source code. To rebuild the shell you will need the Zip Studio 2.0 API, String Studio 2.0 and VBX Studio 1.2. When you register the Zip Studio Shell source code you will be able to unzip the encrypted ZSHL2SRC.ZIP file thanks to your registration password. See the ZSHELL20.HLP help files for details and agreement.

ZipCancel

void WINAPI ZipCancel(BOOL bCancel); [2.0]

Description

Stop the current ZIP action as soon as possible if <bCancel> is set to TRUE. To use this function you must not use the SLOW compression mode but you must turn on the Background processing mode.

Example

```
OnCancelButton(...)
  {
  ZipCancel( TRUE );
  UnzipCancel( TRUE );
  ...
  }
```

Required DLLs

ZDLL20A

See also...

UnzipCancel

ZipSetMode

BOOL WINAPI ZipSetMode(int zMode); [2.0]

Description

Set the compression depending on the following constants:

```
#define ZMODE_NORMAL    0
#define ZMODE_FAST      1
#define ZMODE_SLOW      2
#define ZMODE_STORE     3
```

In most cases the ZMODE_NORMAL is the best choice. ZMODE_STORE doesn' t compress any file. If you wish to allow the user to cancel the ZIP process you must not choose the ZMODE_SLOW mode.

Example

```
ZipSetMode( ZMODE_NORMAL );
AddFileToZip( ...
```

Required DLLs

ZDLL20A

See also...

ZipCancel

ZipSplit

int WINAPI ZipSplit(HWND hParent, LPCSTR szZipName, LPCSTR szDestDir, long lFirstSize, long lNextSize, BOOL bDiskettePause, LPCSTR szTitle, LPCSTR szMsg); [2.0]

Description

Split an existing ZIP file. <hParent> is the window handle to use for notification messages; <szZipName> is the ZIP file name with its full path; <szDestDir> is the directory to use to put the parts; <lFirstSize> is the first part size in bytes and <lNextSize> is the next parts size in bytes; If <bDiskettePause> is set to TRUE the Zip Studio will wait for a new diskette for each part; <szTitle> is the MessageBox title and <szMsg> is the MessageBox main message. <szTitle> and <szMsg> can use %1 and %2: %1 will be replaced with the current part number and %2 with the number of parts. The function returns one of the following value:

Constant	Value	Means
SPLIT_OK	0	Split ok.
SPLIT_NOFILE	1	the ZIP file doesn' t exist.
SPLIT_NOMEM	2	not enough memory.
SPLIT_IOERROR	3	an IO errors occured.
SPLIT_TOOBIG	4	there are more than 10 parts.
SPLIT_ACCESSDENIED	5	Acess denied.
SPLIT_USERABORT	6	The user cancelled the split operation.
SPLIT_FILETOOSMALL	7	There is nothing to do.

Also, if <hParent> is a valid HWND, this window will receive the following notification messages:

Constant	Value	WParam	LParam
ZN_SPLITTING	WM_USER + 61	Part Size in KBytes	Current Part Name.
ZN_SPLITTED	WM_USER + 62	Part Number	Part Size in bytes

Example

```
ZipSplit( m_hWnd, "C:\TEST\MYZIP.ZIP", "A:\", 720000L, 720000L, TRUE, "Disk %1/%2", "Please insert a diskette %1 in A:\" );
```

Required DLLs

ZDLL20A

See also...

ZipJoin

GetZList

int WINAPI GetZList(LPCSTR szZipName, register HWND hListWnd, unsigned int iFrom, int iSortMode, BOOL bBackGnd); [2.0]

Description

This function will build the <szZipName> ZIP file files list. <szZipName> is an existing ZIP file name with its path. <hListWnd> is your system ListBox HWND. This listbox must not have the LBS_SORT style and must be visible. To hide this ListBox, move to 0,0,0,0 with the SDK <MoveWindow> routine for instance. If you can' t do this, use the Zip Studio VBX. <iFrom> is a zero based index from which to Zip Studio will build the list: If you want to retrieve all the ZIP items, give 0. <iSortMode> is one of the following:

```
#define ZSORT_BYFULLNAME    0
#define ZSORT_BYNAME        1
#define ZSORT_BYDIR         2
#define ZSORT_BYDATE        3
#define ZSORT_BYSIZE        4
#define ZSORT_BYRATE        5
#define ZSORT_NONE          6
```

If <bBackGnd> is set to TRUE, the a background process will be used: In some cases you must disable this options. The function returns TRUE if the ZIP file list contains all the ZIP files informations, -1 if an error occured, or, a zero based index if the ZIP file is too big (e.g. the datas were truncated). If the returned value is an index, you can call the function again with the <iFrom> parameter set to this index. To speed up the operation (specially with big files), set <iSortMode> to ZSORT_NONE. When the Zip list is built, you must use the GetZipItem routine to retrieve the informations.

Example

```
register HWND hSysListBox = ::GetDlgItem( hWnd, IDC_SYSLISTBOX );
GetZList( "C:\TEST\MYZIP.ZIP", hSysListBox , 0, ZSORT_NONE,  FALSE );
...
```

Required DLL

UZDLL20

See also...

GetZipItem
UnzipSortZList

GetZipItem

BOOL WINAPI GetZipItem(register HWND hListWnd, int iIndex, ZIPITEM FAR* zItem); [2.0]
typedef struct TAGZIPITEM
{
char szName[50];
char szDir[260];
BOOL bIsDir;
BOOL bIsEncrypted;
int iMethod;
int iDay;
int iMonth;
int iYear;
int iHour;
int iMin;
long lOriginalSize;
int iRate;
} ZIPITEM; [2.0]

Description

You must call this function once the Zip List is built (thanks to the GetZList routine). <hListWnd> is the ListBox handle you gave to GetZList , <iIndex> is the zero based index for the item you want to retrieve the informations (it must lbe less than the CountFileInZip value); <zItem> is a pointer to a ZIPITEM object. If you use VB, the zItem is specified in ZIP.BAS so, you don' t need to declare this object. The functions fills the <zItem> with the file informations. <iYear> doesn' t include any century specification and is in the 00-99 range, other ZIPITEM value are obvious. The function returns TRUE if successful, FALSE if not.

Example

```
ZIPITEM zItem;
register HWND hSysList = ::GetDlgItem( hWnd, IDC_SYSLISTBOX );
char szMsg[512];
if ( GetZList( hWnd, "C:\\MYZIP.ZIP", hSysList, 0, ZSORT_NONE, FALSE ) == 0
)
{
if ( GetZipItem( hSysList, 0, (ZIPITEM FAR*)&zItem ) )
 {
 wsprintf( szMsg, "File: %s, Method: %i, Original Size: %l, Date:
 %i/%i/19%i %i:%i...", zItem.szName, zItem.iMethod, zItem.lOriginalSize,
 zItem.iMonth, zItem.iDay, zItem.iYear, zItem.iHour, zItem.iMin );
 MessageBox( hWnd, szMsg, "ZIPITEM informations", MB_OK );
 }
}
```

Required DLL

UZDLL20

See also...

GetZList

UnzipSortZList

BOOL WINAPI UnzipSortZList(register HWND hListWnd, int iSortMode, BOOL bBackGnd); [2.0]

Description

This function will sort the Zip files list. Before you can call this function, you must build the Zip list thanks to the GetZList routine. As the GetZList rountine can sort the list, this function is useful when you wish to sort the list once again (e.g. with a different mode). <hListWnd> is the ListBox HWND you gave to GetZList, iSortMode is one of the following constant and if <bBackGround> is set to TRUE, the process will run in the background. The function returns TRUE if successful, FALSE if not. Sorting big ZIP files list is not recommended since it can last more than 10 secondes.

```
#define ZSORT_BYFULLNAME    0
#define ZSORT_BYNAME        1
#define ZSORT_BYDIR         2
#define ZSORT_BYDATE        3
#define ZSORT_BYSIZE        4
#define ZSORT_BYRATE        5
#define ZSORT_NONE          6
```

Example

```
ZIPITEM zItem;
register HWND hSysList = ::GetDlgItem( hWnd, IDC_SYSLISTBOX );
char szMsg[512];
if ( GetZList( hWnd, "C:\\MYZIP.ZIP", hSysList, 0, ZSORT_NONE, FALSE ) == 0
)
{
UnzipSortZList( hSysList, ZSORT_BYFULLNAME, TRUE );
...
```

Required DLL

UZDLL20

See also...

GetZList

GetZFileIsEncrypted

BOOL WINAPI GetZFileIsEncrypted(LPCSTR szZIPFileName, LPCSTR szFileName); [2.0]

Description

This function returns TRUE if the file <szFileName> in the ZIP file <szZIPFileName> is encrypted, FALSE if not.

Example

```
if ( GetZFileIsEncrypted("MYZIP.ZIP",szFirstFileInZip))
  MessageBox( hWnd, szFirstFileInZip,"MYZIP.ZIP - Encrypted:", MB_OK);
```

Required DLL

UZDLL20

See also...

CountFileInZip
IsThisFileAZipFile
IsFileUnzipable
GetZFileOriginalSize
GetZFileCompressedSize
GetZFileFullInfos
GetZFileIsDir

UnzipSetPassword

BOOL WINAPI UnzipSetPassword(LPCSTR szPassword); [2.0]

Description

This function tells the Zip Studio to use the <szPassword> to extract or view the encrypted files in the Zip file. If this password is not valid, depending on your SetAskPassword mode, the Zip Studio will display or not the password dialog box. The password must not be longer than 79 characters and is Case sensitive. A NULL password is allowed.

Example

```
SetAskPassword( FALSE );
UnzipSetPassword( "password" );
```

Required DLL

UZDLL20

See also...

SetAskPassword

SetAskPassword

void WINAPI SetAskPassword(BOOL bAsk); [2.0]

Description

The function sets the interactive password mode: If <bAsk> is set to TRUE, the Zip Studio shell will ask the user to give a password for the encrypted file, otherwise the Zip Studio Shell will bypass the encrypted files.

Example

```
SetAskPassword( FALSE );
UnzipSetPassword( "password" );
```

Required DLL

UZDLL20

See also...

UnzipSetPassword

UnzipSetPasswordText

BOOL WINAPI UnzipSetPasswordText(LPCSTR szDialogTitle, LPCSTR szDialogText, LPCSTR szPasswordLabel, LPCSTR szYesBtn, LPCSTR szNoBtn, LPCSTR szNeverBtn); {2.0]

Description

Like the REPLACE dialog box you can change the password dialog box labels. When a new password is required you will receive a ZN_NEWPASSWORD message (if you gave a valid HWND): At this time you can use this function to change the labels: <szDialogTitle> is the password dialog title, <szDialogText> is the main dialog text, <szPasswordLabel> is the 'Password:' label, <szYesBtn>, <szNoBtn> and <szNeverBtn> are the buttons labels. The function returns TRUE if the password dialog box is visible.

Example

```
case ZN_NEX_PASSWORD:
// retrieve the passsword : (LPCSTR)lp from '\t' (if any)
// retrieve the passsword : (LPCSTR)lp to '\t'
UnzipSetPasswordText( "Password", szMsg, "Password:", "&Yes", "&No",
"Ne&ver" );
```

Required DLL

UZDLL20

See also...

ExtractZipFiles
ViewFileFromZip

UnzipCancel

void WINAPI UnzipCancel(BOOL bCancel); [2.0]

Description - Important!

The function will stop the current UNZIP action as soon as possible if <bCancel> is set to TRUE. If you use this function, you must call UnzipCancel(FALSE) each time you call an unzip function to reset the cancel flag. To do this is not necessary with the ZipCancel routine.

Example

```
OnCancelButton(...)
{
ZipCancel( TRUE );
UnzipCancel( TRUE );
}
...

OnView(...)
{
UnzipCancel( FALSE );
....
```

Required DLL

UZDLL20

See also...

ZipCancel

ZipJoin

int WINAPI ZipJoin(HWND hParent, LPCSTR szName, LPCSTR szDestDir, BOOL bSendMsg, LPCSTR szTitle, LPCSTR szMsg); [2.0]

Description

Rebuild a ZIP file from some parts. <hParent> is the window handle to use for notification messages; <szName> is a part file name: A part file name is *.x_x (MYZIP.0_1 for instance). You can give the first part file name but this not compulsory; <szDestDir> is the directory to use to put the new Zip file; If <bSendMsg> is set to TRUE the process will run in the background; <szTitle> is the MessageBox title and <szMsg> is the MessageBox main message. <szTitle> and <szMsg> can use %1 and %2: %1 will be replaced with the current part number and %2 with the number of parts. The function returns one of the following value:

Constant	Value	Means
JOIN_OK	0	Join ok.
JOIN_NOFILE	1	the part file doesn' t exist.
JOIN_NOMEM	2	not enough memory.
JOIN_IOERROR	3	an IO errors occured.
JOIN_ACCESSDENIED	4	Acess denied.
JOIN_USERABORT	5	The user cancelled the split operation.
JOIN_NOTASPLITTEDFILE	7	Bad part file name
JOIN_ZIPEXIST	8	The destination file already exists

Also, if <hParent> is a valid HWND, this window will receive the following notification messages:

Constant	Value	WParam	LParam
ZN_JOINING	WM_USER + 41	Part Size in KBytes	Current Part Name.
ZN_JOINED	WM_USER + 42	Part Number	Part Size in bytes
ZN_JOINDONESIZE	WM_USER + 43	-	Total Size in bytes
ZN_JOINDONENAME	WM_USER + 44	-	The ZIP name

Example

```
ZipJoin( hWnd, "A:\\MYZIP.3_4", "C:\\TEST\\", TRUE, "Diskette %1/%2",
"Insert the diskette %1 in A:\\" );
```

Required DLL

UZDLL20

See also...

ZipSplit

Zip Studio Shell

Overview

The Zip Studio Shell 2.0 is an enhanced Zip Shell for Windows 3.x. The tool is quite easy to use but really powerful: It supports MDI, Drag and Drop, WinFile DDE, Encrypted and multiple parts Zip files... It can runs external tools to use other archives format. It proposes advanced functionalities like the SEARCH or the VIEW commands. Thanks to the Zip Studio 2.0 API this shell is a fast and reliable way to use Zip files with Windows 3.x. And the Zip Studio is a complete Zip solution: Single licence for $25, 10 or 100 licences packs from $3.8, this shell source code and of course the Zip Studio 2.0 API.

Zip Studio Shell source code

The Zip Studio Shell 2.0 Visual C++ MFC 2.5 Source code is available. There are no warranties nor Technical Support regarding this source code. To rebuild the Shell you will need the Zip Studio 2.0 API, VBX Studio 1.2 and String Studio 2.0. There are no source code available for these 3 other toolkits.

Agreement - Very important!

The source code is provide for private use: You must not rebuild the same shell then to distribute it! That' s why we give the source code to allow you to customize this shell for your company for instance. Notice that in this case you are encouraged to register a licence pack even if you use a customized version, since your registration will help us to make some enhancement to this software... Also, you can use the source code in your commercial software as long as your product is not our Zip Studio Shell, e.g. you build a different software. When you register the source code you agree with this agreement, otherwise you must not register the Zip Studio Shell source code.

See also...

The Zip Studio Shell help file (ZSHELL20.HLP) for details.

New properties and Events [2.0]

The ZIP Studio 2.0 VBX performs direct calls to the Zip Studio API. That' s why you will find a complete help with the API functions descriptions ('Related function(s)').

Common properties

Action [2.0]

Type: Short
Related function(s): AddFileToZip , ZipDeleteFiles , ZipRepare , ExtractZipFiles
This is the main special property. Thanks to this property you will choose an action (an operation) like ADDING a file to the ZIP file (1), DELETING a file from the ZIP file (2), EXTRACTING the ZIP file contents (3), VIEWING a file (4), FIXING the ZIP file (5), DELETING the ZIP comment (6), .BUILD THE ZIP LIST (7), SORTING THE ZIP LIST (8), CANCELING AN ACTION (9), JOINING A FILE (10), SPLITTING A FILE (11), or TESTING THE ZIP FILE (12). Of course you must set all the necessary properties before you call this one (specially the <ZipName>, <FileNameOrMask> and <UnzipDestDir>). When the operation is done an <ActionResult> notification message will be sent out but you can also read the <ActionResult> property. Regarding the new actions SPLIT and JOIN, you must fill the <Split...> and <Join...> properties and these action don' t use <ActionResult> but <JoinResult> and <SplitResult>. Also, when you build the ZIP list, the result will be available in the <ZListResult> property.

szPassword [2.0]

Type: String
Related function(s): AddFileToZipExtractZipFiles , UnzipSetPassword
This is the password you will use to build encrypted files or to unzip encrypted files. To disable this encryptation, give "".

Zip properties

CompressionMode [2.0]

Type: Short
This property will set the compression mode (method) for zipping. Default is NORMAL.

Unzip and information properties

AskForPassword [2.0]

Type: Bool
Related function(s): SetAskPassword
Set this property to TRUE if you wish the Zip Studio asks the user for a password if the password is not valid for a file.

Join properties

JoinDestDir [2.0]

Type: String
Related function(s): ZipJoin
Give the destination directory for the ZIP file. If you give '?' a dialog box will be displayed.

JoinMsg [2.0]

Type: String
Related function(s): ZipJoin
This is the Join message box tiltle. %1 will be replaced with the diskette number and %2 with the number of diskettes.

JoinResult [2.0]

Type: Short
Related function(s): ZipJoin
The last join operation result.

JoinSrcFile [2.0]
Type: String
Related function(s): ZipJoin
This is the name of a part file (*.?_?). If you give '?' a dialog box will be displayed.

JoinTitle [2.0]
Type: String
Related function(s): ZipJoin
This is the Join message box tiltle. %1 will be replaced with the diskette number and %2 with the number of diskettes.

Split properties

SplitDestDir [2.0]
Type: String
Related function(s): ZipSplit
Give the destination directory for the part files. If you give '?' a dialog box will be displayed.

SplitDiskettePause [2.0]
Type: BOOL
Related function(s): ZipSplit
If set to TRUE the Zip Studio Shell will wait for a new diskette.

SplitFirstBlockSize [2.0]
Type: Long
Related function(s): ZipSplit
This is the size (in bytes) of the first part file.

SplitMsg [2.0]
Type: String
Related function(s): ZipSplit
This is the Split message box tiltle. %1 will be replaced with the diskette number and %2 with the number of diskettes.

SplitNextBlockSize [2.0]
Type: Long
Related function(s): ZipSplit
This is the size (in bytes) of the next parts files.

SplitResult [2.0]
Type: Short
Related function(s): ZipSplit
The last split operation result.

SplitSrcFile [2.0]
Type: String
Related function(s): ZipSplit
This is the name of an existing ZIP file . If you give '?' a dialog box will be displayed.

SplitTitle [2.0]
Type: String
Related function(s): ZipSplit
This is the Split message box tiltle. %1 will be replaced with the diskette number and %2 with the number of diskettes.

ZList properties

ZListDay [2.0]
Type: Short
Related function(s): GetZList , GetZipItem
The last modification date (day) of the ZList current item (selected thanks to the <ZListIndex> property).

ZListDir [2.0]
Type: String
Related function(s): GetZList , GetZipItem
The relative directory of the ZList current item (selected thanks to the <ZListIndex> property).

ZListIndex [2.0]
Type: Short
Related function(s): GetZList , GetZipItem
Before you can read the following <ZList..> properties, you must go to a specified item in the list (0 indexed). We used this method for speed because we read the Zip Item just one time, then we read the <ZList...> property.

ZListIsDir [2.0]
Type: BOOL
Related function(s): GetZList , GetZipItem
TRUE if the ZList current item (selected thanks to the <ZListIndex> property) is a directory.

ZListIsEncrypted [2.0]
Type: BOOL
Related function(s): GetZList , GetZipItem
TRUE if the ZList current item (selected thanks to the <ZListIndex> property) is encrypted.

ZListFrom [2.0]
Type: Short
Related function(s): GetZList
If you use a too big ZIP file, maybe the ZList doesn' t contain all the files informations. In this case the <ZListResult> is the last read file. So, you can give this index to this property to retrieve the next files infos (to build a new ZList beginning with the item <ZListFrom>).

ZListHour [2.0]
Type: Short
Related function(s): GetZList , GetZipItem
The last modification time (hour) of the ZList current item (selected thanks to the <ZListIndex> property).

ZListMethod [2.0]
Type: Short
Related function(s): GetZList , GetZipItem GetZCompressMethod
The compression method of the ZList current item (selected thanks to the <ZListIndex> property).

ZListMin [2.0]
Type: Short
Related function(s): GetZList , GetZipItem
The last modification time (minute) of the ZList current item (selected thanks to the <ZListIndex> property).

ZListMonth [2.0]
Type: Short
Related function(s): GetZList , GetZipItem
The last modification date (month) of the ZList current item (selected thanks to the <ZListIndex> property).

ZListName [2.0]
Type: String
Related function(s): GetZList , GetZipItem
The name of the ZList current item (selected thanks to the <ZListIndex> property). This name doesn' t include any

path and can be a blank string if the ZList item is a directory.

ZListOriginalSize [2.0]
Type: Long
Related function(s): GetZList , GetZipItem
The original (not compressed) size in bytes of the ZList current item (selected thanks to the <ZListIndex> property). To retrieve the compressed size, you will use this value with the <ZListRate> property.

ZListRate [2.0]
Type: Short
Related function(s): GetZList , GetZipItem
The compressio rate of the ZList current item (selected thanks to the <ZListIndex> property).

ZListResult [2.0]
Type: Short
Related function(s): GetZList
There are 3 cases: The ZIP file can' t be read, the property value is -1. The Zip file was successfully loaded and all items were read, the property value is 0. The ZIP file is too big, this property value is the last read file. So, you can give this index to the <ZListFrom> property to retrieve the next files infos (to build a new ZList beginning with the item <ZListFrom>).

ZListYear [2.0]
Type: Short
Related function(s): GetZList , GetZipItem
The last modification date (year (00-99)) of the ZList current item (selected thanks to the <ZListIndex> property).

ZListZortMode [2.0]
Type: Short
Related function(s): GetZList, UnzipSortZList
You must use this property to set the Zip List sort mode. To make the change call the <Action> property.

ZFile properties

ZFileIsEncrypted [2.0]
Type: Bool (array)
Related function(s): GetZFileIsEncryptedr
Set to TRUE if the file X in a ZIP file (0 indexed) is encrypted.

Events

Encrypt [2.0]
Params: iRate as integer, FileName as String
Related function(s): AddFileToZip
Sent out when the Zip Studio encrypt a file. <iRate> is the rate and <FileName> is the current file name.

JoinDoneName [2.0]
Params: ZipName as String
Related function(s): ZipJoin
The new ZIP file is ready to be used. <FileSize> is the new ZIP file name.

JoinDoneSize [2.0]
Params: FileSize as Long
Related function(s): ZipJoin
The new ZIP file is ready to be used. <FileSize> is the new ZIP file length.

Joined [2.0]
Params: PartNumber as integer, FileSize as Long
Related function(s): ZipJoin
Sent out when a part is joinded. <PartNumber> is the piece file number, <FileSize> is the new ZIP file length.

Joining [2.0]
Params: iRate as integer, PieceName as String
Related function(s): ZipJoin
Sent out when the Zip Studio is joining a ZIP file. <iRate> is the rate (it can be wrong if the parts have different sizes) and <PieceName> is the current file name.

NetRead, NetWrite [2.0]
Params: iRate as integer, FileName as String
Related function(s): AddFileToZip
Sent out when the Zip Studio uses a temporary file to zip on a network. <iRate> is the rate and <FileName> is the current file name.

NewPassord [2.0]
Params: FileName as String, Password as String
Related function(s): ExtractZipFiles , SetAskPassword
Sent out when a new password is required.

Splitted [2.0]
Params: PartNumber as integer, FileSize as Long
Related function(s): ZipSplit
Sent out when a part is done. <PartNumber> is the piece file number, <FileSize> is the new part file length.

Splitting [2.0]
Params: Size as integer, PieceName as String
Related function(s): ZipSplit
Zip Studio is splitting the Zip file. <Size> is the part size in KBytes, PieceName is the part name.

Test [2.0]
Params: bOk as integer, FileName as String
Related function(s): IsThisFileAZIPFile
Sent out when the Zip Studio tests a ZIP file. <bOk> is TRUE if the file is OK and <FileName> is the current file name.

See also...

- VBX Overview
- Events
- VBXREG registration utility
- The VB sample

How to build a ZIP files list?

There are 3 steps to build a Zip files list (using the ZList fast method):

1/ Build (or declare an HWND for) a ListBox. This listbox will have the WS_VISIBLE stype but not the LBS_SORT style. To hide this control you can build this control outside of your form for instance. This listbox is called the SysList.

2/ Build the ZIP list using this ListBox handle and the GetZList function.

3/ Now, retrieve the items values thanks to the GetZipItem function.

Regarding the VBX, you don' t have to build a ListBox!

How to cancel an action?

To cancel a Zip or Unzip action you must call ZipCancel and/or UnzipCancel. You must not call these functions with critical operation (like the ZipDeleteFiles), and you can' t cancel a ZIP operation if you choose the ZMODE_SLOW.

If you use UnzipCancel, you must reset the UNZIP Cancel flag each time (and before) you call an UNZIP function.

Zipping on a network...

When you use ZIP files on a network there is nothing special but the AddFilesToZIP treatment.

To do this, the Zip Studio will use a temporary file on your local computer. That' s why a diskless computer won' t be able to zip on a network since the Zip Studio can' t build the temporary file. Also, the Zip Studio understand drives higher than D: (e.g. F: to Z:) are mapped drives. If they are not mapped you will just lose some treatment times with the temporary files. However, if you use a network drive mapped as C:, you won' t be able to zip into this file.

Using encrypted ZIP files...

Encrypted Zip files are Zip files with one or more encrypted files which requires a password to unzip or view them.

Building an encrypted Zip file...

To add a file into a Zip file with a password protection , you must give this password to the AddFilesToZIP routine. This password is case sensitive and must be smaller than 80 characters. If your Zip files already contains some (not encrypted) files, these one will be encrypted during the process; That' s why, in case you want to build a Zip file with some encrypted files and some 'normal' files, you must begin with the password , then you will remove the password to add the not encrypted files. To use a zip file with more than one password you will do the same.

Using multiple parts ZIP files...

Multiple parts Zip files are a set of parts named *.?_? you must Join them before you can read the Zip file. These parts are not compatible with PKWare' s PKZIP 2.04g. If you need to use PKZIP with the Zip Studio multiple parts Zip files, you must join them with the Zip Studio before you can use PKZIP with that Zip file.

Splitting the files...

The ZipSplit routine will split a Zip file into 2 to 10 parts file. To join these files, you can use the Zip Studio ZipJoin routine, but, if you give these files to someone else it is still possible to use them: To join these parts file you can use the DOS command COPY <file> /B + ... <zipfile> or, maybe you can use another tool (or shareware) to join these parts files. Before you can use these parts files you must join them.

Joining the files...

The ZipJoin routine will join a multiple parts Zip file. To join these files, you can use the Zip Studio ZipJoin function, but, if you give these files to someone else it is still possible to use them: To join these parts file you can use the DOS command COPY <file> /B + ... <zipfile> or, maybe you can use another tool (or shareware) to join these parts files. Before you can use these parts files you must join them.