Issue Date: FoxTalk April 2000

Zipping and Unzipping for Clients

Richard David Hodder
rickhodder@msn.com

Compression software is a staple tool on the developer's belt. The DynaZIP-AX ActiveX controls from Inner Media, Inc. give developers programmatic control over creating and reading ZIP files. In this article, Richard David Hodder introduces you to the capabilities of these controls. Next month, he'll dig deeper into the controls and provide a small framework for their use.

PKZIP and PKUNZIP by PKWARE, Inc. have become household names in the software industry. These are DOS command-line products, and using them to create and manipulate ZIP files in VFP is fraught with difficulty.

First of all, you must use the RUN command to shell out to DOS to run the programs. The DOS window appears on the toolbar. There are memory issues and PIF settings that might need to be tweaked on a user-by-user basis. Occasionally, DOS windows will remain onscreen after a DOS executable finishes. And sometimes it's necessary to put in calls to INKEY() to wait for the DOS program to finish.

Second, to find out the success of the attempt to zip or unzip, the log file must be parsed to determine whether everything was successful.

Third, particularly when zipping or unzipping a large amount of information, it's important to give the user visual feedback as to the status of the zipping/unzipping. Windows users expect (dare I say demand?) something akin to a thermometer during waiting periods. There's no way I know of to hook into PKZIP/PKUNZIP for status information, so unless your clients enjoying watching the DOS box, they'll have to wait until the process is finished. This leaves the door open for inquisitive users to click on the DOS box and cause havoc! To paraphrase the Warner Brothers cartoon representation of John Steinbeck's *Of Mice and Men*, "I'm gonna click on the pretty DOS window, George, and then I'm gonna hug it and squeeze it.…"

Finally, there's a price, minimal though it is. PKZIP and PKUNZIP must be on each user's machine, and there's no royalty-free distribution of PKZIP and PKUNZIP—it's necessary to purchase a copy of PKZIP for each user.

DynaZIP

Inner Media, Inc. has created a product called DynaZIP, which includes (among other things) ActiveX controls that provide developers with programmatic control over zipping and unzipping ZIP files.

Forms of product

DynaZIP is packaged in three different ways:

1. DynaZIP Complete 16
2. DynaZIP Complete 32
3. DynaZIP-AX

The "complete" products include ActiveX controls, Static DLLs, OCXs, and VCLs. The AX product has only the 32-bit ActiveX controls. This article focuses on the DynaZIP-AX product.

Evaluation copies

Some vendors offer what are called "crippled" versions of products for evaluation. "Crippling" refers to the fact that you can't test all of the functionality of the product. In addition to crippling, vendors often put what are called evaluation "time bombs" into the demo. These time bombs aren't destructive; they merely prevent you from using the demo product after an evaluation period has elapsed. Both of these techniques are used to generate interest in a product without giving away the farm.

Inner Media, Inc. provides free evaluation copies of its software on its Web site (http://www.innermedia.com/html/free_demos.htm). The evaluation copies aren't crippled and have no time bombs in them. The evaluation period for the product is 30 days, but you're on the honor system. To quote the company's Web site, "Of course, we are relying on your honor to treat these copyrighted works properly, and to come back within 30 days and purchase a legal copy once you've decided it's right for you. You trust us with your precious data, and we'll trust you to honor our license agreement." The evaluation copy even includes a huge manual in PDF format and code samples in several different development languages. Inner Media goes a step further, providing 30 days of technical support via e-mail. This company wants your business!

Features

DynaZIP-AX is chock full of wonderful functionality:

- **Zipping capability**: Create new ZIP files, add or delete items from existing ZIP files, and freshen or update existing ZIP files. ZIP from memory to an item in a ZIP file, or ZIP from memory to memory.
- **Unzipping capability**: Extract all or just some items from a ZIP file; get information about each item in a ZIP file. Extract all or a portion of a ZIP file item to memory.
- **Long filename support**: Automatically supports long filenames in the Windows 95/98 and Windows NT environments. Includes options for zipping and unzipping to systems that don't support long filenames.
- **UNC support**: You can use the Universal Naming Convention (UNC) instead of the drive letter when specifying a path or filename.
- **True multi-threading support**: Contains true multi-processed and multi-threaded capable DLLs/components.
- **Active Server Pages (ASP) support**: Contains sample code showing how to use the windowless ActiveX COM components with ASP.
- **Renaming capability**: Rename items/files during zipping/unzipping operations.
- **Multiple volume support (disk-spanning)**: Create ZIP files that span more than one disk.
- **Encryption/decryption**: Add security to your ZIP files by adding password protection to some or all of the items.
- **International support**: Easily customize/translate all DynaZIP messages that your end user might see.
- **Support for many development languages**: Choose which DynaZIP programming interface to use with your development environment: ActiveX, OCX, DLL, or VCL. Or use the "DZ_EASY" interface, which makes most DynaZIP functions easily available to database languages. You can use DynaZIP with many different programming languages, including Visual Basic, VBScript, Delphi, C/C++, Visual FoxPro, Access, PowerBuilder, Pascal, and other contemporary development environments.
- **Double-byte character set (DBCS) Support**: Develop applications that service most international markets. With DBCS, your programs will easily manage filenames, messages, and so on. in the full range of displayable characters, long or short.
- **Royalty-free distribution**: Distribute DynaZIP ActiveX DLLs with your applications at no extra cost.
- **Diagnostic tools and sample code**: Use any of the many useful sample applications and copy parts of their source code into your own application.
- **Useful documentation**: Includes a full manual in PDF format and online Help.

There are three controls included in DynaZIP-AX: the DynaStat control, the Zip control, and the Unzip control.

DynaStat control

The DynaStat control is an ActiveX thermometer that can be dropped on a form and used to display status. **Figure 1** shows what the DynaStat control looks like in action. It's a fairly simple control with just a few important properties:

- **StatusPercent** (Numeric): This property determines the percentage displayed on the thermometer, as well as how

much of the thermometer bar will be filled in. It should contain a value in the range of 0 to 100. In Figure 1, the StatusPercent is set to 10.

- **StatusText** (Character): This property determines the text to be displayed in the thermometer area of the control. For example, you might display the name of the item currently being compressed. In Figure 1, the StatusText is set to "XYZ.TXT."

- **MarkerColor** (Numeric): This property determines the color used in the thermometer bar.

- **Use3Dflag** (Logical): This property determines whether the control looks three-dimensional or flat.

You can use the DynaStat control in combination with the MajorStatus and MinorStatus callback events to present a different status thermometer while zipping and unzipping.

***Figure 1:** The DynaStat control.*

The blender metaphor

I recently bought a blender. It doesn't have an On/Off switch. It has an Off switch and several blend settings. You want to know the first crucial lesson I learned? *Never* add things to a blender when it's working! Turn it off first, then add what you want, and then choose your blend setting. This metaphor applies to both the Zip and the Unzip controls.

Neither the Zip control nor the Unzip control has a method that kicks off the zipping/unzipping process. Instead, both have a property called ActionDZ that's set to one action out of a list of possibilities (ZIP_ADD, UNZIP_EXTRACT, or similar actions)—the blend settings, if you will. There are two actions named ZIP_NOACTION and UNZIP_NOACTION. These settings don't necessarily represent the social life of developers. They're DynaZIP's equivalent of the "Off" switch. It's a good idea to turn off the controls using the NOACTION actions between changes in actions.

Zip control

Rather than present the entire interface of the Zip control (which is laid out in the documentation), I'll present progressive examples that demonstrate the use of the control, introducing properties as I go.

When I create a ZIP file (with or without DynaZIP), I need to know a few basic things. The first thing is the name of the ZIP file to create. Second is the list of items to add to the ZIP file. Finally, I need to know how to make the Zip control add the specified items to the ZIP file (in other words, how to start the blender). The following code sample demonstrates how to create a ZIP file, called example1.zip, that will contain all files beginning with "File" and ending with a TXT extension in the current directory:

```
*-- Create the Zip control
loZip = CREATEOBJECT("dzactxctrl.dzactxctrl.1")
WITH loZip
  *-- Turn off the blender
  .ActionDZ = ZIP_NOACTION
  *-- Set the name of the ZIP file
  .ZipFile=SYS(5)+CURDIR()+"example1.zip"
  *-- Set the items to put into the ZIP file
  .ItemList = SYS(5)+CURDIR()+"file*.txt"
  *-- Tell the Zip control not to store the
  *-- path of the items in the ZIP file
  .NoDirectoryNamesFlag = .T.

  *-- Tell the Zip control to add
  .ActionDZ= ZIP_ADD
  *-- Check the success of the zip
  llSuccess = (.ErrorCode = ZE_OK)
ENDWITH
```

Very simple! No messy parsing or DOS windows, and I even get an indication of whether there were any problems! Let's cover the properties introduced in this sample:

- **ZipFile** (Character): The name basically says it all: This property should be set to the name of the ZIP file to create or manipulate. When setting this property, it's very important to include a full path.

- **ItemList** (Character): This is the filespec identifying the files you want to include in the ZIP file. As you can see from the sample, wildcard characters (for example, *,?) can be included in the filespec. In fact, multiple filespecs can be included in the ItemList at the same time, each delimited by a space character. For example, I could set the ItemList property to "c:\rick.* c:\article??.doc c:\readme.txt." Notice that, like the ZipFile property, a full path must be included in each filespec entered in this property. What DynaZIP does with the items in the ItemList is determined by the ActionDZ property (the blender setting).

- **NoDirectoryNamesFlag** (Logical): This property determines whether the path to the files in the ItemList is included in the ZIP file. If this is set to .T. (as in the preceding code), the paths won't be included in the ZIP.

- **ErrorCode** (Numeric): This property holds a numeric code that represents whether the last action performed by the Zip control was successful. Any non-zero value in this property signals that something didn't work properly. The Include file for the Zip control has a list of the error codes (all of which have the prefix "ZE_") returned in this property.

The ActionDZ property is the blender setting. It determines the specific action to be taken on the items in the ItemList (among other things). As mentioned before, it has an "Off" setting (ZIP_NOACTION), and it has seven other settings. Two of these (ZIP_MEMTOMEM and ZIP_MEMTOFILESTREAM) don't apply to VFP, so they won't be discussed here. Four of the actions should be very familiar to anyone who uses compression software: Add, Delete, Update, and Freshen (the #DEFINEs for these actions all have a ZIP_ prefix). The ZIP_ADD setting adds the items in the ItemList property to the ZIP file. The ZIP_DELETE setting removes the items in the ItemList from the ZIP file; if this setting had been used in the preceding code, it would have removed file*.txt from the ZIP file. When using the ZIP_UPDATE setting, items in the ItemList that don't already exist in the ZIP file are added, and items that already exist in the ZIP and have a later modification date are replaced (updated). The ZIP_FRESHEN setting replaces an existing item in a ZIP file, but only if it's been modified more recently than the item in the ZIP file.

ZIP_MEMTOZIP

The final ActionDZ setting that I'll discuss is ZIP_MEMTOZIP. Two words: "Way cool!" Employing this setting is like using low-level file functions to add an item to a ZIP file. Simply fill in the ZipString property with the text you'd like to be in the file, then set the ActionDZ property to ZIP_MEMTOZIP. Here's a sample that creates an entry in the ZIP file called "readme.txt":

```
*-- Create the Zip control
loZip = CREATEOBJECT("dzactxctrl.dzactxctrl.1")
WITH loZip
   *-- Turn off the blender
   .ActionDZ = ZIP_NOACTION
   *-- Set the name of the ZIP file
   .ZipFile=SYS(5)+CURDIR()+"memtozip.zip"
   *-- Set the item in the zip that the
   *-- memory (stored in ZipString) will
   *-- be written to
   .ItemList = "readme.txt"

   *-- Set what will get written to readme.txt
   .ZipString = "Here are the latest release notes..."

   .ActionDZ= ZIP_MEMTOFILE
ENDWITH
```

This can save a step in a process. Rather than creating a readme.txt file and then adding it to the ZIP file, I can just add the readme.txt file directly.

Status

The Zip control allows the program to present a status thermometer while zipping. I can set the caption of the thermometer form by setting the ExtProgTitle property. There's also a numeric property called "ZipSubOptions," which, among other things, can be used to show a status thermometer during the action being performed.

```
*-- Create the Zip control
loZip = CREATEOBJECT("dzactxctrl.dzactxctrl.1")
WITH loZip
  *-- Turn off the blender
  .ActionDZ = ZIP_NOACTION
  *-- Set the name of the ZIP file
  .ZipFile=SYS(5)+CURDIR()+"example2.zip"
  *-- Set the items to put into the ZIP file
  .ItemList = SYS(5)+CURDIR()+"file*.txt"
  *-- Tell the Zip control not to store the
  *-- path of the items in the ZIP file
  .NoDirectoryNamesFlag = .T.
```

```
  *--Set the thermometer title
  .ExtProgTitle = "Zipping In Progress"
  *-- Use the thermometer included with DynaZIP-AX
  .ZipSubOptions = ZSO_EXTERNALPROG

  *-- Tell the Zip control to add
  .ActionDZ= ZIP_ADD
  *-- Check the success of the zip
  llSuccess = (.ErrorCode = ZE_OK)
ENDWITH
```

A number of #DEFINEs are associated with the ZipSubOptions property, and they all begin with the prefix ZSO. The ZipSubOptions property is a set of binary flags that can be configured by adding together the #DEFINEs for the required functionality. For example, a #DEFINE called ZSO_EXTPROGCANCEL tells the DynaZIP thermometer to surface a Cancel button, and another called ZSO_RESET_ARCHIVED resets the archive bit on any file zipped. If I want to use the DynaZIP thermometer with a cancel button and have it reset the archived bit, I'd replace the line beginning with ".ZipSubOptions=" with the following line of code:

```
.ZipSubOptions = ZSO_EXTERNALPROG  + ;
                 ZSO_EXTPROGCANCEL + ;
                 ZSO_RESET_ARCHIVED
```

The ZipSubOptions property can also be used to write a log of the zipping activity, ignore long file names, and perform many other functions.

MultiVolume (disk spanning)

Disk spanning refers to creating ZIP files that extend across several diskettes or removable media. The numeric property MultiVolume configures the type of disk spanning used. If this property is set to zero, no spanning will occur. A number of #DEFINEs are associated with the MultiVolume property, and they all begin with the prefix MV_. The MultiVolume property is a set of binary flags that can be configured by adding together the #DEFINEs for the required functionality.

```
loZip.MultiVolume = MV_USEMULTI + ;
                    MV_FORMAT   + ;
                    MV_LOWDENSE
```

The preceding line of code tells the ZIP control to use disk spanning and to format each disk at low density.

Other properties of interest

Here are descriptions of a few more properties. (The DynaZIP manual covers many others.)

- **EncryptCode** (Character): Filling in this property allows the encryption password (up to 65 characters) to be set for the items being zipped. In addition to filling in this property, you must set the EncryptFlag property to .T. to enable encryption.

- **Comment** (Character): Filling in this property allows a user to set the comment of the ZIP file. In addition to filling in this property, the user must set the AddCommentFlag property to .T. to use this function during the ZIP action.

- **RecurseFlag** (Logical): Setting this property to .T. is the equivalent of telling PKZIP to recurse through the subdirectories when building the contents of the ZIP file.

- **FixFlag** (Logical): Ever get the message that the ZIP file is damaged, along with instructions to use PKFIX to attempt to fix it? Setting this flag to .T. will attempt to fix the ZIP file. If the attempt to fix the ZIP file fails, DynaZIP has another flag called "FixHarderFlag" (I kid you not!) that can be used to try to resurrect your information.

Unzip control

Rather than present the entire interface of the Unzip control (which is laid out in the documentation), I'll present progressive examples that demonstrate the use of the control, introducing properties as I go.

When I unzip a ZIP file (with or without DynaZIP), I need to know a few basic things. The first thing is the name of the ZIP file to extract from. Second is the filespec for the items to extract from the ZIP file. Third is the destination—where those extracted files should be put. Finally, I need to know how to make the Unzip control start extracting (in other words, how to start the Unzip blender). The following code sample demonstrates how to extract all files beginning with "file" and ending with a TXT extension, moving them from the ZIP file called example1.zip into a subdirectory of the current directory, called output:

```
*-- Create the Unzip control
loUnZip = CREATEOBJECT("duzactxctrl.duzactxctrl.1")
WITH loUnZip
  *-- Turn off the blender
  .ActionDZ = UNZIP_NOACTION
  *-- Set the name of the ZIP file
  .ZipFile=SYS(5)+CURDIR()+"example1.zip"
  *-- Set which files to extract
  .FileSpec="file*.txt"
  *-- Set the directory to put the
  *-- extracted items
  .Destination = SYS(5)+CURDIR()+"output"

  *-- Tell the Zip control to extract
  .ActionDZ = UNZIP_EXTRACT
  *-- Check the success of the unzip
  llSuccess = (.ErrorCode = UE_OK)
ENDWITH
```

Once again, very easy! Let's cover the properties introduced in this sample.

- **ZipFile** (Character): The name says it all: This property should be set to the name of the ZIP file to manipulate. When setting this property, it's very important to include a full path.

- **FileSpec** (Character): This is the filespec identifying the files that you want to extract in the ZIP file. As you can see from the sample, wildcard characters (for example, *,?) can be included. In fact, multiple filespecs can be included in the FileSpec property at the same time, each delimited by a space character. For example, I could set the FileSpec property to "rick.* article??.doc." What DynaZIP does with the items in FileSpec is determined by the ActionDZ property (the blender setting).

- **Destination** (Character): This property determines the location (path and directory) to which the items in FileSpec will be extracted. When setting this property, it's very important to include a full path.

- **ErrorCode** (Numeric): This property holds a numeric code that indicates whether the last action performed by the Unzip control succeeded. Any non-zero value in this property signals that something didn't work right. The Include file for the ZIP control has a list of the error codes (all of which have the prefix "UE_") returned in this property.

Whereas the primary focus of the ActionDZ actions in the ZIP control is on updating a ZIP file, only two of the 11 actions available to the Unzip control pertain to extracting files (UNZIP_EXTRACT and UNZIP_FILETOMEM). As with the ZIP control, there are two actions (UNZIP_MEMTOMEM and UNZIP_MEMTOFILESTREAM) that don't apply to VFP and thus won't be discussed here. A description of the actions appears in the sidebar titled "ActionDZ Settings for Unzip."

UNZIP_FILETOMEM

More "Way cool!" This action is the converse to ZIP_MEMTOFILE. This action allows you to read sections of an item in a ZIP file, as if you were using low-level file functions. The following code reads the first 20 characters of the readme.txt item in memtozip.ZIP, then displays a message box showing the extracted text. At no time during the extracting of this ZIP file did the readme.txt item leave the ZIP file:

```
*-- Create the Unzip control
loUnZip = CREATEOBJECT("duzactxctrl.duzactxctrl.1")
WITH loUnZip
  *-- Turn off the blender
  .ActionDZ = UNZIP_NOACTION
  *-- Set the name of the ZIP file
  .ZipFile = SYS(5)+CURDIR()+"memtozip.zip"
  *-- Set which files to extract
  .FileSpec = "readme.txt"
  *-- Specify how much text to read
  *-- from the readme.txt file
  .UnzipStringSize = 20
  *-- Specify the character offset that
  *-- determines where to start reading
  *-- the contents of readme.txt.
  *-- 0 = first character in the file
  .UnzipStringOffset = 0

  *-- Tell the Zip control to extract
  .ActionDZ = UNZIP_FILETOMEM
  *-- Display the string in a message box
```

```
  MESSAGEBOX(.UnZipString)
ENDWITH
```

Assuming that you ran the ZIP_MEMTOFILE code sample, you'll see a message box that displays the text "Here are the latest."

Status

Displaying a status thermometer with the Unzip control is nearly identical to the approach used with the Zip control. The caption of the thermometer form can be set with the ExtProgTitle property. Instead of ZipSubOptions, Unzip uses a numeric property called "UnZipSubOptions," which, among other things, can be used to show a status thermometer during the action being performed.

A number of #DEFINEs are associated with the UnZipSubOptions property, and they all begin with the prefix USO. The UnZipSubOptions property is a set of binary flags that can be configured by adding together the #DEFINEs for the required functionality. If I want to use the DynaZIP thermometer with a cancel button and have it reset the archived bit, I'd set UnZipSubOptions in the following manner:

```
.UnZipSubOptions = USO_EXTERNALPROG  + ;
                   USO_EXTPROGCANCEL + ;
                   USO_RESET_ARCHIVED
```

The UnZipSubOptions property can also be used to write a log of the unzipping activity, ignore long file names, and much more.

Other properties of interest

Here are descriptions of a few more properties. (The DynaZIP manual covers many others.)

- **DecryptCode** (Character): When items in a ZIP file are encrypted, you must set this property to the password required to decrypt them. The string can be up to 65 characters in length. In addition to filling in this property, it's necessary to set the DecryptFlag property to .T. to enable decryption.

- **DiagnosticFlag**: Setting this flag to .T. will cause the Unzip control to write out an ASCII text log file of all operations to and from the Unzip control. The log file, named DYNAZIP.LOG, will be located in your Windows directory. The information written out includes the values of the different properties of the control. This is very useful for troubleshooting problems during development.

- **OverwriteFlag**: This is like a SET SAFETY OFF when unzipping items. The Unzip control won't prompt you to overwrite files if this property is set to .T.

- **TestFlag**: Setting this flag to .T. puts the Unzip control into a test mode that goes through all of the motions of unzipping, including extracting the files, but the files are never written to disk. This is fantastic for verifying that each item in the ZIP file can be uncompressed without an error.

Things to watch out for

Repeat this mantra: "Use full paths." ActiveX controls don't recognize VFP's current directory setting. If you execute the code to create a ZIP file, and the ZIP file isn't where you expect it to be or it doesn't appear to have been updated, check to make sure that you included a full path in the ZipFile property. The same can happen anywhere you specify a filespec (ItemList, FileSpec, Destination, or similar specs).

One of the first things that I do when I unwrap a new ActiveX control is to subclass it. You can subclass the Zip and Unzip controls just fine, but once you do, you can't instantiate the subclass using CREATEOBJECT()! It must be hosted on a form. I'll leave a description of this struggle for the next article.

If you decide you can't wait until next month to explore some of the more advanced features, I'll give you one warning. If you use any of the callback events, you should set _VFP.AutoYield to .F. before setting the ActionDZ properties; otherwise, the callback events won't fire.

I've created several include files (.H files) that contain the #DEFINEs used by the DynaZIP controls (and used in code samples in this article). The files DZ32.H, DZ.H, and DUZ.H are available in the accompanying **Download file**. The file DZ32.H has #INCLUDEs for both the DZ.H and DUZ.H files. If you aren't sure which file to include, choose DZ32.H and you'll have everything.

My impressions of DynaZIP-AX

I think that this is an amazing product, one that should be on the "Bat-belt" of any developer. It's obvious that a lot of thought went into the creation of this product. The inclusion of support for multi-threaded capabilities, DBCS, UNC, and international concerns shows a focus on keeping pace with currently evolving technology and the global programming marketplace.

Stay tuned! Next month's article will delve deeper into the product, focusing on the features that provide hooks into customization and internationalization, as well as the ability to change the name and other information of items while zipping and unzipping.

Sidebar: ActionDZ Settings for Unzip

UNZIP_COUNTALLZIPMEMBERS

P>Returns the total number of items in the ZIP file to the ReturnCount property. This count can be affected by the USO_IGNORELONGNAMES bit of the UnZipSubOptions member.

UNZIP_GETNEXTZIPINFO

Retrieves information about the next item in the ZIP file. This information is placed in the properties whose names begin with "zi_," including zi_attr (file attributes), zi_cMethod (method of compression), zi_cOption (related to zi_cMethod), zi_cPathType (whether a path was stored with this item), zi_crc_32 (CRC value), zi_cSize (compressed size), zi_DateTime (datetime stamp), zi_FileName (file name), zi_index (zero-based item number within ZIP file), and zi_oSize (uncompressed size).

To retrieve information about all items in the ZIP file, first use the UNZIP_COUNTALLZIPMEMBERS action to find the number of items. Then use the UNZIP_GETNEXTZIPINFO action that many times.

UNZIP_COUNTNAMEDZIPMEMBERS

Retrieves the number of items in the ZIP file that match the FileSpec property. This action returns the number of counted items in the ReturnCount property. Other properties can affect the final number of items that match the FileSpec.

UNZIP_GETNEXTNAMEDZIPINFO

Retrieves information from the ZIP file about the next item that matches the FileSpec property. This information is placed in the properties whose names begin with "zi_" (see the UNZIP_GETNEXTZIPINFO entry for descriptions). Other properties can affect the final number of selected items.

This is the companion to the UNZIP_COUNTNAMEDZIPMEMBERS action. To retrieve information about all items in the ZIP file that match a particular FileSpec, first use the UNZIP_COUNTNAMEDZIPMEMBERS action to find the number of items. Then use the UNZIP_GETNEXTNAMEDZIPINFO action that many times.

UNZIP_GETCOMMENTSIZE

Retrieves the number of characters in the comment of the ZIP file. This is used to find the size of a comment, allowing you to allocate the appropriate space to read back the comment. This action returns the number of characters in the ReturnCount field.

UNZIP_GETCOMMENT

Retrieves a comment string from the ZIP file. This action uses the value of ReturnCount as a limit of how many characters may be written into the ReturnString property. The number of characters actually retrieved (from 0 to 2,047) is then returned in ReturnCount. This is the companion to the UNZIP_GETCOMMENTSIZE action. To retrieve the whole comment associated with a ZIP file, first use the UNZIP_GETCOMMENTSIZE action to find out how many characters are in the comment. Then use the UNZIP_GETCOMMENT action to retrieve the comment.

UNZIP_GETINDEXEDZIPINFO

Retrieves information about an item (specified by UnZIPIndex) in the ZIP file. The information retrieved is placed in the properties whose names begin with "zi_" (see UNZIP_GETNEXTZIPINFO for descriptions).

UNZIP_EXTRACT

Extracts (uncompresses) selected items from the ZIP file to the directory specified by the Destination property.

UNZIP_FILETOMEM

Extracts (uncompresses) a portion of the selected item and writes it in the string property UnZIPString. When using this function, specify only a single item (without wildcards) in FileSpec, or specify the item using the UnZIPIndex property.

The program UNZIPFXN.PRG is available in the accompanying **Download file.** In it, you'll find functions that demonstrate the use of these actions.