Zip for 4D

****

This plug-in for 4D allow you to programmatically read and write ZIP archives. Both reading and writing uses iterative approach.

When not registered, Zip4D works for 30 minutes. After this trial period, Zip4D will ignore all commands and will return an error -1.

This plug-in uses 'zlib' general-purpose compression library version 1.2.1, November 17th, 2003 Copyright (C) 1995-2003 Jean-loup Gailly and Mark Adler.

Common (auxiliary) routines:

- register the plug-in (ZIP Register)
- convert Date & Time to/from Unix time-stamp [seconds since 1.1.1970] (ZIP 4DDateTime To Time, ZIP TIME TO 4DDATETIME)
- compress/decompress BLOB - internally used for ZIP entries (ZIP Compress BLOB, ZIP Decompress BLOB)

ZIP file reader routines:

- open a ZIP archive (ZIP Open File, ZIP Open BLOB)
- navigate through the archive (ZIP Go To First Entry, ZIP Go To Next Entry, ZIP Go To Entry)
- get information on current ZIP entry (ZIP Get Entry Properties, ZIP Current Entry)
- extract current entry (ZIP Calculate Entry, ZIP Extract Entry, ZIP Extract Into BLOB)
- extract rest of the archive to disk (ZIP Calculate Rest Of Archive, ZIP Extract Rest Of Archive)
- close the archive (ZIP Close)

ZIP file writer routines:

- create a ZIP archive (ZIP Create File, ZIP Create BLOB)
- add an entry to the archive (ZIP Add Entry, ZIP Add BLOB, ZIP Copy)
- add folder hierarchy to the archive (ZIP Calculate Hierarchy, ZIP Add Hierarchy)
- save the archive (ZIP Save File, ZIP Save BLOB) - you can use ZIP Close if you are aborting the archive creation - archive will be invalid

GZip file routines:

- ZIP Open GZip File
- ZIP Read GZip File
- ZIP Create GZip File
- ZIP Write GZip File
- ZIP Close GZip File

Comments to current implementation:

- there is no possibility to update ZIP archives - you have to do it in 4D (using ZIP Copy command)
- only deflation method is supported
- there is no line-endings conversion for text files - just the bit in the archive is set
- there is virtually no file name conversion - ZIP archive (AFAIK) should use iso-8859-1 encoding
- if “show progress window” flag is set, Zip for 4D will display its own progress window.
- there is no support for long names truncation
  - on Mac OS supporting HFS+ API, long names are used (even under Mac OS 9, where Finder does _not_ support long names)
  - on Mac OS not supporting long names, error -37 will be thrown (name too long)
- on Windows the plug-in will fail to receive some information about files with names beginning with dot (e.g. ".DS_Store") - it seems that the Altura is trying to open a driver instead of a file – such files will not have the Mac extra info attached
- there is no check if expanded entry overwrites existing file(s), developer can use 4D callback methods to implement this
- there is no check if someone is trying to zip currently created archive into itself, ...
- error reporting (problem identification) is very limited (generally, error -1 and 2)

*** auxiliary routines ***

ZIP Register (SerialNumber; MacKey; WinKey) -> Result

->	SerialNumber	C_LONGINT	
->	MacKey	C_STRING(24)	
->	WinKey	C_STRING(24)	
<-	result	C_LONGINT	

Register the plug-in. In beta version, this call is ignored.
Result is one of:

0	not registered
1	registered

NOTE: In versions older than 0.9.7, this call was implemented as procedure (no result value), not as function.

ZIP 4DDateTime To Time (Date; Time; convertToUTC) -> UnixTime

->	Date	C_DATE	
->	Time	C_TIME	
->	convertToUTC	C_LONGINT	0 = no, 1 = yes (date & time are local - convert to UTC)
<-	UnixTime	C_LONGINT	

Convert 4D's date & time to Unix time. If convertToUTC is one, resulting time will be converted to UTC (Universal Coordinated Time).

ZIP TIME TO 4DDATETIME (UnixTime; Date; Time; convertToLocalTime)

->	UnixTime	C_LONGINT	
<-	Date	C_DATE	
<-	Time	C_TIME	
->	convertToLocalTime	C_LONGINT	0 = no, 1 = yes (UnixTime is in UTC, convert to local time)

Convert Unix time to 4D's date & time.
ZIP entry times are always in UTC (in this implementation, not necessarily in the ZIP archive)

ZIP Compress BLOB (uncompressed; compressed; makeGZip; level; strategy; memLevel; CRC) -> error

->	uncompressed	C_BLOB	BLOB to compress
<-	compressed	C_BLOB	resulting compressed BLOB
->	makeGZip	C_LONGINT	0 = no (raw compressed data), 1 = yes (GZip header, CRC at end ==> same as GZip file [.gz] on Unix)
->	level	C_LONGINT	0..9, -1 for default of 6 (0 = fastest, 9 = best)
->	strategy	C_LONGINT	0..2, -1 for default of 0 0 normal 1 Huffman only 2 filtered
->	memLevel	C_LONGINT	1..9, -1 for default of 8 (hash bits: value of 8 means use 15 bits in hash table)
<-	CRC	C_LONGINT	valid only if makeGZip = 0
<-	error	C_LONGINT	

Compress BLOB in memory. If makeGZip is zero, it is up to developer to save the size & CRC for later verification.

ZIP Decompress BLOB (compressed; uncompressed; isGZip; CRC) -> error

->	compressed	C_BLOB	compressed BLOB
<-	uncompressed	C_BLOB	resulting uncompressed BLOB
->	isGZip	C_LONGINT	0 = no, 1 = yes (GZip header, CRC at end ==> same as GZip file)
<-	CRC	C_LONGINT	valid only if isGZip = 0
<-	error	C_LONGINT	

Decompress BLOB in memory. If isGZip is zero, it is up to developer to verify the size & CRC.

*** ZIP archive reader routines ***

ZIP Open File (fileName; rootPath; callbackName; cbFlags; NumberOfEntries; ZipComment) -> result

->	fileName	C_STRING(255)	name of the ZIP archive to open
->	rootPath	C_STRING(255)	optional full path to a directory for extraction
->	callbackName	C_STRING(31)	optional name of the 4D method to call
->	cbFlags	C_LONGINT	bit 0 - show progress window, bit 1 - ask for passwords
<-	NumberOfEntries	C_LONGINT	number of entries in the archive
<-	ZipComment	C_STRING(255)	archive's comment
<-	result	C_LONGINT	positive: archiveReference, negative: error

Open existing archive stored in a file for reading.

ZIP Open BLOB (archive; rootPath; callbackName; cbFlags; NumberOfEntries; ZipComment) -> result

<->	archive	C_BLOB	ZIP archive to open
->	rootPath	C_STRING(255)	optional full path to a directory for extraction
->	callbackName	C_STRING(31)	name of the 4D method to call
->	cbFlags	C_LONGINT	bit 0 - show progress window, bit 1 - ask for passwords
<-	NumberOfEntries	C_LONGINT	number of entries in the archive
<-	ZipComment	C_STRING(255)	archive's comment
<-	result	C_LONGINT	positive: archiveReference, negative: error

Open existing archive stored in a BLOB for reading. The routine will clear passed BLOB (archive) and its size will be zero at return!

ZIP Go To First Entry (archiveReference) -> error

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Open File/ZIP Open BLOB
<-	error	C_LONGINT	51 = end of archive

Rewind to the first entry in the ZIP archive.

ZIP Go To Next Entry (archiveReference) -> error

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Open File/ZIP Open BLOB
<-	error	C_LONGINT	51 = end of archive

Skip to next entry in the ZIP archive.

ZIP Go To Entry (archiveReference; entryNumber) -> error

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Open File/ZIP Open BLOB
->	entryNumber	C_LONGINT	in range 1 to NumberOfEntries
<-	error	C_LONGINT	51 = end of archive

Skip to specified entry in the ZIP archive.
NOTE: This function first appeared in Zip4D 0.9.3.

ZIP Get Entry Properties (archiveReference; Name; Size; CTime; MTime; ATime; ExtAttrib; IntAttrib; IntFlags; MadeBy; ComprMethod; ComprLevel; ComprSize; CRC; Comment; MacName; Creator; Type) -> error

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Open File/ZIP Open BLOB
<-	Name	C_TEXT	name of the entry as stored in the archive (e.g. using forward slashes on all platforms)
<-	Size	C_LONGINT	uncompressed size
<-	CTime	C_LONGINT	creation date (UnixTime in UTC)
<-	MTime	C_LONGINT	modification date (UnixTime in UTC)
<-	ATime	C_LONGINT	last access date (UnixTime in UTC)
<-	ExtAttrib	C_LONGINT	external attributes (MS-DOS attributes in low word, Unix attributes in high word)
<-	IntAttrib	C_LONGINT	internal attributes 0 binary 1 ascii
<-	IntFlags	C_LONGINT	internal flags bit 0 entry is encrypted bit 1..3 deflated: 1 for level >= 8 2 for level = 2 3 for level = 1 bit 4 has data descriptor at the end
<-	MadeBy	C_LONGINT	version in low byte (& 0x00ff) 10 version 10 (original PKZip) 20 version 20 (PKZip 2.0.4) platform in high byte (& 0xff00) 0 MS-DOS 7 MacOS 11 Win32
<-	ComprMethod	C_LONGINT	compression method (only STORED & DEFLATED are supported) 0 STORED 1 SHRUNK 2..5 REDUCED 6 IMPLODED 7 TOKENIZED 8 DEFLATED 9 ENHDEFLATED 10 DCLIMPLODED
<-	ComprLevel	C_LONGINT	compression level 0..9 (0 = fastest, 9 = best)
<-	ComprSize	C_LONGINT	compressed size
<-	CRC	C_LONGINT	
<-	Comment	C_STRING(255)	entry's ZIP comment
<-	MacName	C_STRING(31)	stored Macintosh original name
<-	Creator	C_STRING(4)	stored Macintosh file creator
<-	Type	C_STRING(4)	stored Macintosh file type
<-	error	C_LONGINT	

Get [all] properties of current entry in ZIP archive. For directories, the Name ends with a '/'.

ZIP Get Current Entry (archiveReference) -> result

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Open File/ZIP Open BLOB
<-	result	C_LONGINT	positive: current entry number, negative: error

Get current position (entry number) in the archive.
NOTE: This function first appeared in Zip4D 0.9.3.

ZIP Calculate Rest Of Archive (archiveReference; Flags; Count; Size) -> error

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Open File/ZIP Open BLOB
->	Flags	C_LONGINT	bitfield
<-	Count	C_LONGINT	number of entries (files & directories)
<-	Size	C_LONGINT	total [uncompressed] size of entries
<-	error	C_LONGINT	

Pre-flight all remaining entries [including current one] of the archive to get the number & size of files to be decompressed. Needed for progress only.

ZIP Extract Rest Of Archive (archiveReference; Flags; Password) -> error

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Open File/ZIP Open BLOB
->	Flags	C_LONGINT	bitfield
->	Password	C_STRING(255)	optional password to use
<-	error	C_LONGINT	

Extract all remaining entries [including current one] to disk.

ZIP Calculate Entry (archiveReference; Flags; Count; Size) -> error

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Open File/ZIP Open BLOB
->	Flags	C_LONGINT	bitfield
<-	Count	C_LONGINT	number of entries (files & directories)
<-	Size	C_LONGINT	total [uncompressed] size of entries
<-	error	C_LONGINT	

Pre-flight current entry of the archive to get the number & size of files to be decompressed. Needed for progress only.
NOTE: This function first appeared in Zip4D 0.9.3.

ZIP Extract Entry (archiveReference; Flags; Password; FileNameToUse) -> error

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Open File/ZIP Open BLOB
->	Flags	C_LONGINT	bitfield
->	Password	C_STRING(255)	optional password to use
->	FileNameToUse	C_STRING(255)	optional file name to use (override name stored in archive)
<-	error	C_LONGINT	

Extract current entry in ZIP archive into a file.

ZIP Extract Into BLOB (archiveReference; file; Password) -> error

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Open File/ZIP Open BLOB
<-	file	C_BLOB	extracted file
->	Password	C_STRING(255)	optional password to use
<-	error	C_LONGINT	

Extract current entry in ZIP archive into a BLOB.

ZIP Close (archiveReference) -> error

<->	archiveReference	C_LONGINT	result of [successful] call to ZIP Open File/ZIP Open BLOB/ZIP Create File/ZIP Create BLOB archiveReference is cleared (set to zero) if successful if archiveReference is result of call to ZIP Create File/ZIP Create BLOB, the archive is invalid => use ZIP Save to finish the archive creation
<-	error	C_LONGINT	

Close ZIP archive [close file, release all memory used, ...].

*** ZIP archive writer routines ***

ZIP Create File (fileName; rootPath; callbackName; cbFlags) -> result

->	fileName	C_STRING(255)	name of the archive to create
->	rootPath	C_STRING(255)	optional full path to a "root" directory for compression - this path will be stripped from all file names
->	callbackName	C_STRING(31)	name of the 4D method to call
->	cbFlags	C_LONGINT	bit 0 - show progress window bit 12 – create Mac SEA bit 13 – create Win SEA
<-	result	C_LONGINT	positive: archiveReference, negative: error

Create a ZIP archive in a file.
NOTE: The ability to create a SEA (Self Extracting Archive) first appeared in Zip4D 0.9.1.

ZIP Create BLOB (rootPath; callbackName; cbFlags) -> result

->	rootPath	C_STRING(255)	optional full path to a "root" directory for compression - this path will be stripped from all file names
->	callbackName	C_STRING(31)	name of the 4D method to call
->	cbFlags	C_LONGINT	bit 0 - show progress window bit 13 – create Win SEA
<-	result	C_LONGINT	positive: archiveReference, negative: error

Create a ZIP archive in a BLOB.
NOTE: The ability to create a SEA (Self Extracting Archive) first appeared in Zip4D 0.9.1.

ZIP Add Entry (archiveReference; fileName; ComprLevel; Flags; Password; Comment) -> error

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Create File/ZIP Create BLOB
->	fileName	C_STRING(255)	name of the file to compress or name of the directory to store
->	ComprLevel	C_LONGINT	compression level 0..9, -1 for default of 6 (0 = fastest, 9 = best)
->	Flags	C_LONGINT	bitfield
->	Password	C_STRING(255)	optional password to encrypt with
->	Comment	C_STRING(255)	optional comment to store
<-	error	C_LONGINT	

Add an entry into the ZIP archive. For directories, ComprLevel & Password are ignored.

ZIP Calculate Hierarchy (archiveReference; fileName; Flags; NumberOfFiles; SizeOfFiles) -> error

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Create File/ZIP Create BLOB
->	fileName	C_STRING(255)	name of the file/directory to calculate sizes
->	Flags	C_LONGINT	bitfield
<-	NumberOfFiles	C_LONGINT	number of entries (files & directories)
<-	SizeOfFiles	C_LONGINT	total [uncompressed] size of entries
<-	error	C_LONGINT	

Pre-flight the hierarchy to get the number & size of files to be compressed. Needed for progress only.

ZIP Add Hierarchy (archiveReference; fileName; ComprLevel; Flags; Password) -> error

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Create File/ZIP Create BLOB
->	fileName	C_STRING(255)	name of the file/directory to compress
->	ComprLevel	C_LONGINT	compression level 0..9, -1 for default of 6 (0 = fastest, 9 = best)
->	Flags	C_LONGINT	bitfield
->	Password	C_STRING(255)	optional password to encrypt with
<-	error	C_LONGINT	

Add a whole hierarchy to the ZIP archive.

ZIP Add BLOB (archiveReference; file; Flags; Name; CTime; MTime; ATime; ExtAttrib; IntAttrib; ComprLevel:L; Password; Comment; Creator; Type; ComprSize; CRC) -> error

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Create File/ZIP Create BLOB
->	file	C_BLOB	document to compress
->	Flags	C_LONGINT	bitfield
->	Name	C_TEXT	name of the entry as to be stored in the archive (e.g. using forward slashes on both platforms)
->	CTime	C_LONGINT	creation date (UnixTime in UTC)
->	MTime	C_LONGINT	optional modification date (UnixTime in UTC)
->	ATime	C_LONGINT	optional last access date (UnixTime in UTC)
->	ExtAttrib	C_LONGINT	external attributes (MS-DOS attributes in low word, unix attributes in high word)
->	IntAttrib	C_LONGINT	internal attributes 0 binary 1 ascii 2 unknown
->	ComprLevel	C_LONGINT	compression level 0..9, -1 for default of 6 (0 = fastest, 9 = best)
->	Password	C_STRING(255)	optional password to encrypt with
->	Comment	C_STRING(255)	optional comment to store
->	Creator	C_STRING(4)	optional Macintosh file creator
->	Type	C_STRING(4)	optional Macintosh file type
<-	ComprSize	C_LONGINT	compressed size
<-	CRC	C_LONGINT	
<-	error	C_LONGINT	

Add a BLOB into the ZIP archive. For directories (name ending with '/'), ComprLevel & Password are ignored.

ZIP Copy (newArchiveRef; oldArchiveRef) -> error

->	newArchiveRef	C_LONGINT	result of [successful] call to ZIP Create File/ZIP Create BLOB
->	oldArchiveRef	C_LONGINT	result of [successful] call to ZIP Open File/ZIP Open BLOB
<-	error	C_LONGINT	

Copy current entry from one ZIP archive to another.

ZIP Save File (archiveReference; ZipComment) -> error

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Create File archiveReference is cleared (set to zero) if successful
->	ZipComment	C_STRING(255)	optional archive's comment
<-	error	C_LONGINT	

Save (and close) the ZIP archive.

ZIP Save BLOB (archiveReference; ZipComment; archive) -> error

->	archiveReference	C_LONGINT	result of [successful] call to ZIP Create BLOB archiveReference is cleared (set to zero) if successful
->	ZipComment	C_STRING(255)	optional archive's comment
<-	archive	C_BLOB	created archive
<-	error	C_LONGINT	

Save (and close) the ZIP archive.

Flags bitfield meaning:

for zipping:

Zip_Default	0	use MacBinary if needed (Mac only), store UTC times, store Mac info
Zip_DataForkOnly	1	store data forks only (Mac only - always true on Win)
Zip_UseMacBinary	2	always use MacBinary encoding (Mac only)
Zip_JunkPath	4	do not store path (store file names only)
Zip_Ascii	8	treat as text file
Zip_Binary	16	treat as binary file
Zip_NoUTC	32	do not store UTC time extra fields
Zip_NoMacInfo	64	do not store Mac extra fields
Zip_StoreReadOnly	128	store read-only flags (mark file as RO if it is read-only)
Zip_NoDirectories	256	do not put directory entries into the archive
Zip_NoInvisibleFiles	512	ignore invisible files/directories (HIDDEN on Win, invisible on Mac)
Zip_StoreMacComments	1024	store Mac Finder comments as ZIP entry comments (Mac only)
Zip_StoreFullNamesAsComments	2048	store full path (platform native) as ZIP entry comments
Zip_DoNotStoreIntoArchive	1073741824	do not put the files into the ZIP archive, only headers are put into the archive (useful for file hierarchy traversal)

for unzipping:

Zip_Default	0	decode MacBinary if needed (on Win, only data fork will be extracted)
Zip_DataForkOnly	1	extract data forks only (always true on Win)
Zip_DoNotExpandMacBinary	2	do not decode MacBinary
Zip_JunkPath	4	do not use path (flat file names only, no directories)
Zip_UseReadOnly	128	use read-only flags (make file RO if marked as such in archive)
Zip_NoDirectories	256	ignore directory entries in the archive
Zip_NoInvisibleFiles	512	ignore invisible files/directories (HIDDEN on Win, invisible on Mac)
Zip_UseMacComments	1024	use ZIP entry comments as Mac Finder comments (Mac only)

Callback 4D method arguments and return values:

$0	C_LONGINT	<-	result
	Zip_CB_Continue	0	continue operation
	Zip_CB_Skip	-1	skip current entry
	Zip_CB_Abort	-128	abort completely (e.g. user clicked "Abort" in progress window)
	any_other_value		abort completely
$1	C_TEXT	->	file name [Zip format for Zip_CB_GetFile, Zip_CB_PutName, native otherwise]
$2	C_LONGINT	->	operation
	Zip_CB_GetFile	0	Unzip this ZIP entry? Zip entry passed in $1 in Zip format Called before extracting a zip entry Return value in $0
	Zip_CB_GetName	1	Get name for this file/directory Name of file/directory passed in $1 in Native OS format Called before extracting a zip entry Return new name of file/directory in $5 or keep the name if $5 empty
	Zip_CB_GetPassword	2	Get password for this file Name of file passed in $1 in Native OS format Called before extracting a zip entry Return password for file in $5. If empty string is returned and “ask for passwords” bit is set, plug-in will display its own request dialog.
	Zip_CB_Descend	3	Descend this directory? Name of directory passed in $1 in Native OS format Called at directory entry when compressing a hierarchy Return value in $0
	Zip_CB_PutFile	4	Zip this file/directory? Name of file/directory passed in $1 in Native OS format Called before compressing file Return value in $0
	Zip_CB_PutName	5	Name for this ZIP entry Name of file/directory passed in $1 in Zip format Called before adding new entry into archive Return new name of file/directory in $5 or keep the name if $5 empty
	Zip_CB_Unzipping	6	progress unzipping file Use this event to implement custom progress
	Zip_CB_Gathering	7	progress calculating sizes for zipping (ZIP Calculate Hierarchy) or unzipping (ZIP Calculate Rest Of Archive) Use this event to implement custom progress
	Zip_CB_Zipping	8	progress zipping file Use this event to implement custom progress
$3	C_LONGINT	->	processed part of current file (for Zip_CB_Unzipping and Zip_CB_Zipping), total files so far (for Zip_CB_Gathering)
$4	C_LONGINT	->	size of current file (for Zip_CB_Unzipping and Zip_CB_Zipping), total size so far (for Zip_CB_Gathering)
$5	C_TEXT	<-	name of the current file (for Zip_CB_GetName, Zip_CB_PutName), password (for Zip_CB_GetPassword)

*** GZip file routines ***
NOTE: Functions for streaming access to GZip (.gz) files first appeared in Zip4D 0.9.5.
NOTE: Encryption (and argument "password") first appeared in Zip4D 0.9.6.

ZIP Open GZip File (fileName; password) -> result

->	fileName	C_STRING(255)	name of the archive to create
->	password	C_STRING(255)	optional password for encryption
<-	result	C_LONGINT	positive: fileReference, negative: error

Open a GZip file for reading and check the header.
If password is not empty, encryption is used. If the password is incorrect, the stream will encounter compression errors (some ZIP Read GZip File call will fail).

ZIP Read GZip File (fileReference; blob; size) -> result

->	fileReference	C_LONGINT	result of [successful] call to ZIP Open GZip File
->	blob	C_BLOB	data storage; must not be empty (is never resized)
->	size	C_LONGINT	number of bytes to read
<-	result	C_LONGINT	positive: number of bytes read, negative: error

Read data from a GZip file. If size is zero, blob's size is used.

ZIP Create GZip File (fileName; level; strategy; memLevel; password) -> result

->	fileName	C_STRING(255)	name of the archive to create
->	level	C_LONGINT	0..9, -1 for default of 6 (0 = fastest, 9 = best)
->	strategy	C_LONGINT	0..2, -1 for default of 0 0 normal 1 Huffman only 2 filtered
->	memLevel	C_LONGINT	1..9, -1 for default of 8 (hash bits: value of 8 means use 15 bits in hash table)
->	password	C_STRING(255)	optional password for encryption
<-	result	C_LONGINT	positive: fileReference, negative: error

Create and open a GZip file for writing, write GZip header.
Encryption is used if password is not empty.

ZIP Write GZip File (fileReference; blob; size) -> result

->	fileReference	C_LONGINT	result of [successful] call to ZIP Create GZip File
->	blob	C_BLOB	data storage; must not be empty (is never resized)
->	size	C_LONGINT	number of bytes to write
<-	result	C_LONGINT	positive: number of bytes written, negative: error

Write data into a GZip file. If size is zero, blob's size is used.

ZIP Close GZip File (fileReference) -> error

<->	fileReference	C_STRING(255)	result of [successful] call to ZIP Open GZip File or ZIP Create GZip File
<-	error	C_LONGINT	

For reader (file opened with call to ZIP Open GZip File):
- check the CRC and decompressed size
- close the file
- set fileReference to zero

For writer (file opened with call to ZIP Create GZip File):
- write the CRC and decompressed size
- close the file
- set fileReference to zero