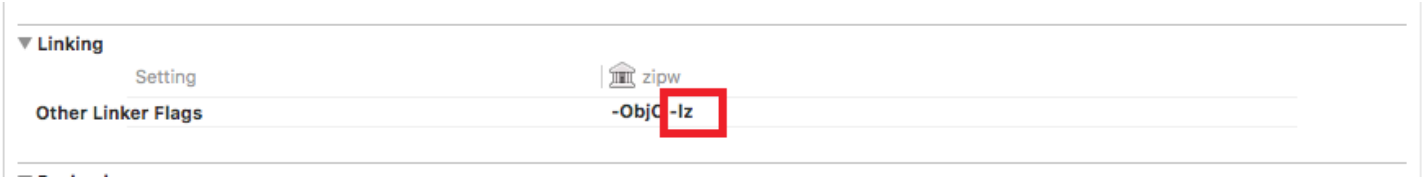
Zip / gzip Multiplatform Native Plugin.

Thank you for purchasing this shared library for Android, iOS*, WindowsPhone8.1,Windows 10 Universal, Windows, OSX, Linux & WebGL**.

The scope of this library is to compress/decompress zip/gzip archives and buffers on Android, iOS, WindowsPhone8.1, Windows10 Universal, Windows, OSX,Linux.
The ios and OSX libraries are compiled as universal. That means that they will support 32 and 64 bit builds.

(**bitcode** enabled iOS plugins are provided in the **plugins/ios/bitcode folder**.)
The Windows and Linux libraries are compiled for x86 and x86_64 build modes.
The Android lib is compiled for armeabi, armeabi-v7a, x86 and x86_64.
The Windows Phone 8.1 lib is compiled for ARM and x86.
THE WSA lib is compiled for ARM, x86 and x86_64.
WSA SDK10 and SDK8.1 is supported with NetCore for Unity version 5.2 and higher.
(ios: if you want to compile for watchOS,tvOS or simulator, extract in the Plugins/iOS folder and replace the desired plugin.)

***iOS/tvOS/watchOS compilation requires to add the -lz linking flag at Build Settings-> Linking-> Other Linker flags on xcode.**



**WebGL supports only compression/decompression of zlib/gzip buffers.
*** bz2 method is not available for MacOS/iOS/tvOS/watchOS.
**** encyption/decryption is not available for WSA due to certification reasons.

FEATURES:

Fast zip/gzip compression and decompression with a clean and simple interface. Very easy to use.

The plugin is about **7x times faster** in compression speed and **3x times faster** in decompression speed compared to SharpZipLib.

- compress/decompress buffers to/from zlib/gzip streams.
- recursive directory compression/decompression.
- compress/decompress single files.
- encryption / decryption. (not available for WSA)
- append files to existing zip archives.
- compress a buffer and write it or append it to a zip archive.
- get file and size info of all the files or a specific file from a zip archive.
- extract a single file out of a zip archive.
- decompress a file of a zip archive to a byte buffer.
- delete an entry in a zip archive.
- replace an entry in a zip archive.
- get progress of extraction when the zip archive has multiple files.
- bz2/zlib compression-decompression methods. (bz2 method not available for MacOS/iOS/tvOS/watchOS)

*- Linux, iOS, Android, MacOSX, WSA & Windows can treat **buffers as files**. That means if you have a file in www.bytes you can perform operations directly on the buffer.*
For Android this is very useful since you can decompress from Streaming Assets without copying to Persistent data path.
!!! The plugin will not decompress __MACOSX folders, files starting with ._ and files with 0 bytes size!!!

INSTRUCTIONS:

If you want to run a small example, compile and run the testScene.
It will download a small zip file and it will perform all the functions the lib provides.
See the lzip.cs file for more comments and error codes.
In your project include in the Plugins folder the plugins you want to use and the lzip.cs file and call the appropriate functions as described below and shown in the demo scene.

THE FUNCTIONS:

int getTotalFiles(string zipArchive, byte[] FileBuffer = null);

A function that returns the total number of files in a zip archive.

ZipArchive : the zip to be checked
FileBuffer : A buffer that holds a zip file. When assigned the function will read from this buffer and will ignore the filePath.
ERROR CODES
: -1 = failed to access zip archive
: any number>0 = the number of files in the zip archive

---

int getTotalEntries(string zipArchive, byte[] FileBuffer = null);

A function that will return the total entries in a zip arcive. (files + folders)

ERROR CODES
: -2 = failed to access zip archive
: any number>0 = the number of entries in the zip archive

---

long getFileInfo(string zipArchive, string path = null, byte[] FileBuffer = null);

This function fills the Lists with the filenames and file sizes that are in the zip file.

Returns : the total size of uncompressed bytes of the files in the zip archive
zipArchive : the full path to the archive, including the archives name. (/myPath/myArchive.zip)
path *: this is no longer used. It is kept for a while for backwards* compatibility.
FileBuffer : A buffer that holds a zip file. When assigned the function will read from this buffer and will ignore the filePath.
ERROR CODES : -1 = Input file not found
: -2 = Could not get info

---

int getEntrySize(string zipArchive, string entry, byte[] FileBuffer = null);

A function that returns the uncompressed size of a file in a zip archive.

zipArchive : the zip archive to get the info from.
Entry : the entry for which we want to know it uncompressed size.
FileBuffer : A buffer that holds a zip file. When assigned the function will read from this buffer and will ignore the filePath.

---

bool entryExists(string zipArchive, string entry, byte[] FileBuffer = null);

A function that tells if an entry in zip archive exists.

Returns true or false.
ZipArchive : the zip archive to get the info from.
entry : the entry for which we want to know if it exists.
FileBuffer : A buffer that holds a zip file. When assigned the function will read from this buffer and will ignore the filePath.

---

bool compressBuffer(byte[] source, ref byte[] outBuffer, int levelOfCompression);

A function that compresses a byte buffer to a zlib stream compressed buffer. Provide a reference buffer to write to. This buffer will be resized.

source : the input buffer
outBuffer : the referenced output buffer
levelOfCompression : (0-10) recommended 9 for maximum (10 is highest but slower and not zlib compatible)
ERROR CODES : true = success
: false = failed

---

byte[] compressBuffer(byte[] source, int levelOfCompression);

A function that compresses a byte buffer to a zlib stream compressed buffer. Returns a new buffer with the compressed data.

source : the input buffer
levelOfCompression : (0-10) recommended 9 for maximum (10 is highest but slower and not zlib compatible)
ERROR CODES : a valid byte buffer = success
: null = failed

---

int compressBufferFixed(byte[] source, ref byte[] outBuffer, int levelOfCompression, bool safe = true);

Same as the compressBuffer function, only this function will put the result in a fixed size buffer to avoid memory allocations.
The compressed size is returned so you can manipulate it at will.

Safe : if set to true the function will abort if the compressed resut is larger the the fixed size output buffer.
Otherwise compressed data will be written only until the end of the fixed output buffer.

---

bool decompressBuffer(byte[] source, ref byte[] outBuffer);

A function that decompresses a zlib compressed buffer to a referenced outBuffer. The outbuffer will be resized.

source : a zlib compressed buffer.
outBuffer : a referenced out buffer provided to extract the data. This buffer will be resized to fit the uncompressed data.
ERROR CODES : true = success
: false = failed

---

byte[] decompressBuffer(byte[] source);

A function that decompresses a zlib compressed buffer and creates a new buffer. Returns a new buffer with the uncompressed data.

source : a zlib compressed buffer.
ERROR CODES : a valid byte buffer = success
: null = failed

---

int decompressBufferFixed(byte[] source, ref byte[] outBuffer, bool safe = true);

Ssame as the decompressBuffer function. Only this one outputs to a buffer of fixed which size isn't resized to avoid memory allocations.
The fixed buffer should have a size that will be able to hold the incoming decompressed data.

Returns the uncompressed size.

Safe : if set to true the function will abort if the decompressed resut is larger the fixed size output buffer.
Otherwise decompressed data will be written only until the end of the fixed output buffer.

---

int entry2Buffer(string zipArchive, string entry, ref byte[] buffer, byte[] FileBuffer = null, string password = null);

A function that will decompress a file in a zip archive directly in a provided byte buffer.

zipArchive	: the full path to the zip archive from which a specific file will be extracted to a byte buffer.
entry	: the file we want to extract to a buffer. (If the file resides in a directory, the directory should be included.)
buffer	: a referenced byte buffer that will be resized and will be filled with the extraction data.
FileBuffer	: A buffer that holds a zip file. When assigned the function will read from this buffer and will ignore the filePath.
password	: If the archive is encrypted use a password.
ERROR CODES	: 1 = success
	: -2 = could not find/open zip archive
	: -3 = could not locate entry
	: -4 = could not get entry info
	: -5 = password error
	: -18 = entry has no size
	: -104 = internal memory error

---

int entry2FixedBuffer(string zipArchive, string entry, ref byte[] fixedBuffer, byte[] FileBuffer = null, string password = null);

A function that will decompress a file in a zip archive directly in a provided fixed size byte buffer.

Returns the uncompressed size of the entry.

zipArchive	: the full path to the zip archive from which a specific file will be extracted to a byte buffer.
entry	: the file we want to extract to a buffer. (If the file resides in a directory, the directory should be included.)
buffer	: a referenced fixed size byte buffer that will be filled with the extraction data. It should be large enough to store the data.
FileBuffer	: A buffer that holds a zip file. When assigned the function will read from this buffer and will ignore the filePath.
password	: If the archive is encrypted use a password.
ERROR CODES	: 1 = success
	: -2 = could not find/open zip archive
	: -3 = could not locate entry
	: -4 = could not get entry info
	: -5 = password error
	: -18 = entry has no size
	: -19 = the fixed size buffer is not big enough to store the uncompressed data
	: -104 = internal memory error

---

byte[] entry2Buffer(string zipArchive, string entry, byte[] FileBuffer = null, string password = null));

A function that will decompress a file in a zip archive in a new created and returned byte buffer.

zipArchive	: the full path to the zip archive from which a specific file will be extracted to a byte buffer.
entry	: the file we want to extract to a buffer. (If the file resides in a directory, the directory should be included.)
FileBuffer	: A buffer that holds a zip file. When assigned the function will read from this buffer and will ignore the filePath.
password	: If the archive is encrypted use a password.
ERROR CODES	: non-null = success
	: null = failed

---

bool buffer2File(int levelOfCompression, string zipArchive, string arc_filename, byte[] buffer, bool append=false);

A function that compresses a byte buffer and writes it to a zip file. I you set the append flag to true, the output will get appended to an existing zip archive.

levelOfCompression	: (0-9) recommended 9 for maximum.
zipArchive	: the full path to the zip archive to be created or append to.
arc_filename	: the name of the file that will be written to the archive.
buffer	: the buffer that will be compressed and will be put in the zip archive.
append	: set to true if you want the output to be appended to an existing zip archive.

comment : an optional comment for this entry.
password : an optional password to encrypt this entry.
useBz2 : set to true if you want bz2 compression instead of zlib. (Except MacOS/iOS/tvOS/watchOS)
ERROR CODES : true = success
: false = failed

---

int delete_entry(string zipArchive, string arc_filename);

A function that deletes a file in a zip archive. It creates a temp file where the compressed data of the old archive is copied except the one that needs to be deleted.

After that the old zip archive is deleted and the temp file gets renamed to the original zip archive.
You can delete directories too if they are empty.

zipArchive : the full path to the zip archive
arc_filename : the name of the file that will be deleted.
ERROR CODES : 1 = success
: -1 = failed to open zip
: -2 = failed to locate the archive to be deleted in the zip file
: -3 = error copying compressed data from original zip
: -4 = failed to create temp zip file.

---

int replace_entry(string zipArchive, string arc_filename, string newFilePath, int level = 9, string comment=null, string password = null, bool useBz2=false);

A function that replaces an entry in a zip archive with a file that lies in a path. The original name of the archive will be used.

zipArchive : the full path to the zip archive
arc_filename : the name of the file that will be replaced.
newFilePath : a path to the file that will replace the original entry.
Level : the level of compression of the new entry.
comment : add a comment for the file in the zip file header.
password : set the password to protect this file. (only ascii characters)
useBz2 : use the bz2 compression algorithm. If false the zlib deflate algorithm will be used. (Except MacOS/iOS/tvOS/watchOS)
ERROR CODES
: -1 = could not create or append
: -2 = error during operation
: -3 = failed to delete original entry

---

int replace_entry(string zipArchive, string arc_filename, byte[] newFileBuffer, int level = 9, string password = null, bool useBz2 = false);

A function that replaces an entry in a zip archive with a buffer. The original name of the archive will be used.

zipArchive : the full path to the zip archive
arc_filename : the name of the file that will be replaced.
newFileBuffer : a byte buffer that will replace the original entry.
Level : the level of compression of the new entry.
password : set the password to protect this file. (only ascii characters)
useBz2 : use the bz2 compression algorithm. If false the zlib deflate algorithm will be used. (Except MacOS/iOS/tvOS/watchOS)
ERROR CODES
: 1 = success
: -5 = failed to delete the original file
: -6 = failed to append the buffer to the zip

---

int extract_entry(string zipArchive, string arc_filename, string outpath, byte[] FileBuffer = null, int[] proc = null, string password = null);

A function that will extract only the specified file that resides in a zip archive.

zipArchive : the full path to the zip archive from which we want to extract the specific file.
arc_filename : the specific file we want to extract. (If the file resides in a directory, the directory path should be included. Like dir1/dir2/myfile.bin)
--> on some zip files the internal dir structure uses \\ instead of / characters for directories separators. In that case use the appropriate chars that will allow the file to be extracted.

outpath : the full path to where the file should be extracted + the desired name for it.
FileBuffer : A buffer that holds a zip file. When assigned the function will read from this buffer and will ignore the filePath.
Proc : a single item integer array that gets updated with the progress of thedecompression in bytes. (100% is reached when the compressed size of the file is reached.)
password : if needed, the password to decrypt the entry.
ERROR CODES : -1 = extraction failed
: -2 = could not initialize zip archive.
: -3 = could not locate entry
: -4 = could not get entry info
: -5 = password error
: 1 = success

---

int decompress_File(string zipArchive, string outPath, int[] progress, byte[] FileBuffer = null, int[] proc = null, string password = null);

A function that decompresses a zip file. If the zip contains directories, they will be created.

zipArchive : the full path to the zip archive that will be decompressed.
outPath : the directory in which the zip contents will be extracted.
progress : provide a single item integer array to write the current index of the file getting extracted. To use it in realtime, call this function in a separate thread.
FileBuffer : A buffer that holds a zip file. When assigned the function will read from this buffer and will ignore the filePath.
proc : a single item integer array that gets updated with the progress of the decompression in bytes. (100% is reached when the compressed size of the file is reached.)
password : if needed, the password to decrypt the archive.
ERROR CODES
: -1 = could not initialize zip archive.
: -2 = failed extraction
: 1 = success

---

int compress_File(int levelOfCompression, string zipArchive, string inFilePath,bool append=false, string fileName="", string comment=null, string password = null, bool useBz2 = false);

A function that compresses a file to a zip file. If the flag append is set to true then it will get appended to an existing zip file.

LevelOfCompression : (0-9) recommended 9 for maximum (10 is highest but slower and not zlib compatible)
zipArchive : the full path to the zip archive that will be created
inFilePath : the full path to the file that should be compressed and added to the zip file.
append : set to true if you want the input file to get appended to an existing zip file. (if the zip file does not exist it will be created.)
filename : if you want the name of your file to be different then the one it has, set it here. If you add a folder structure to it, like (dir1/dir2/myfile.bin), the directories will be created in the zip file.
comment : add a comment for the file in the zip file header.
password : set the password to protect this file. (only ascii characters)
useBz2 : use the bz2 compression algorithm. If false the zlib deflate algorithm will be used. (Except MacOS/iOS/tvOS/watchOS)
ERROR CODES
: 1 = success
: -1 = could not create or append
: -2 = error during operation

---

int compress_File_List(int levelOfCompression, string zipArchive, string[] inFilePath, int[] progress = null, bool append=false, string[] fileName=null, string password = null, bool useBz2 = false);

A function that compresses a list of files to a zip file.

LevelOfCompression	: (0-9) recommended 9 for maximum (10 is highest but slower and not zlib compatible)
zipArchive	: the full path to the zip archive that will be created
inFilePath[]	: an array of the full paths to the files that should be compressed and added to the zip file.
progress	: this var will increment until the number of the input files and this are equal.
append	: set to true if you want the input file to get appended to an existing zip file. (if the zip file does not exist it will be created.)
filename[]	if you want the names of your files to be different then the one they have, set it here. If you add a folder structure to it, like (dir1/dir2/myfile.bin), the directories will be created in the zip file.
password	: set the password to protect this file. (only ascii characters)
useBz2	: use the bz2 compression algorithm. If false the zlib deflate algorithm will be used. (Except MacOS/iOS/tvOS/watchOS)
ERROR CODES	
	: 1 = success
	: -1 = could not create or append
	: -2 = error during operation

---

compressDir(string sourceDir, int levelOfCompression, string zipArchive, bool includeRoot = false, string password = null, bool useBz2 = false);

Compress a directory with all its files and subfolders to a zip file (Does not work on WSA 8.1)

sourceDir	: the directory you want to compress
levelOfCompression	: the level of compression (0-9).
zipArchive	: the full path+name to the zip file to be created .
includeRoot	: set to true if you want the root folder of the directory to be included in the zip archive. Otherwise leave it to false.
password	: set the password to protect this file. (only ascii characters)
useBz2	: use the bz2 compression algorithm. If false the zlib deflate algorithm will be used. (Except MacOS/iOS/tvOS/watchOS)

If you want to get the progress of compression, call the getAllFiles function to get the total number of files in a directory and its subdirectories.
The compressDir when called from a separate thread will update the public static int cProgress.
Divide this with the total number of files (as floats) and you have the % of the procedure.

---

int getAllFiles(string Dir); *//(Does not work on WSA 8.1)*

Use this function to get the total files of a directory and its subdirectories.

[Android, iOS, Linux, MacOSX only]
int setFilePermissions(string filePath, string _user, string _group, string _other);

Sets permissions of a file in user, group, other.
Each string should contain any or all chars of "rwx".
Returns 0 on success.

---

bool setEncoding(uint encoding);

Set encoding of file names (read/write) on **Windows and WSA10**

CP_ACP = 0
CP_OEMCP/UNICODE = 1
CP_UTF8 = 65001
CP_WINUNICODE = 1200

---

bool validateFile(string zipArchive, byte[] FileBuffer = null);

A function that will validate a zip archive.

zipArchive : the zip to be checked
FileBuffer : A buffer that holds a zip file. When assigned the function will read from this buffer and will ignore the filePath.
ERROR CODES
: true. The archive is ok.
: false. The archive could not be validated.

DateTime entryDateTime(string zipArchive, string entry, byte[] FileBuffer = null);

A function to get the DateTime of an entry in a zip archive

Returns the date and time of the entry in DateTime format.

zipArchive : the path to a zip archive from which we want to check an entry.
entry : the specific entry we want to get the DateTime of. (If the entry resides in a directory, the directory path should be included. like dir1/dir2/myfile.bin)
FileBuffer : A buffer that holds a zip file. When assigned the function will read from this buffer and will ignore the zipArchive path.
ERROR CODES
: 0 = Cannot open zip Archive
: 1 = entry not found
: 2 = error reading entry

GZIP SECTION

---

int gzip(byte[] source, byte[] outBuffer, int level, bool addHeader = true, bool addFooter = true);

Compress a byte buffer to gzip format.

Returns the size of the compressed buffer.

source : the uncompressed input buffer.
outBuffer : the provided output buffer where the compressed data will be stored (it should be at least the size of the input buffer +18 bytes).
level : the level of compression (0-9)
addHeader : if a gzip header should be added. (recommended if you want to write out a gzip file)
addFooter : if a gzip footer should be added. (recommended if you want to write out a gzip file)

If you add a header and a footer to the buffer, you can write out a file that is a regular gzip file.

---

int gzipUncompressedSize(byte[] source);

Get the uncompressed size from a gzip buffer that has a footer included.

source : the gzip compressed input buffer. (it should have at least a gzip footer).

---

int unGzip(byte[] source, byte[] outBuffer, bool hasHeader = true, bool hasFooter = true);

Decompress a gzip buffer.

Returns the uncompressed size. Negative error code on error.

source : the gzip compressed input buffer.
outBuffer : the provided output buffer where the uncompressed data will be stored.
hasHeader : if the buffer has a header.
hasFooter : if the buffer has a footer.

---

int unGzip2(byte[] source, byte[] outBuffer);

Decompress a gzip buffer. (This function assumes that the gzip buffer has a gzip header !!!)

Returns: the uncompressed size. negative error code on error.

source : the gzip compressed input buffer.
outBuffer : the provided output buffer where the uncompressed data will be stored.

SUPPORT:

For any questions, problems and suggestions please use this email address: elias_t@yahoo.com
Forum: http://forum.unity3d.com/threads/released-zip-native-multiplatfrom-plugin.339482
