This is a plugin based on the brotli library released under the MIT license (see above).

The scope of this library is to compress/decompress brotli archives and buffers on Android, iOS, Windows, OSX & Linux. *WebGl supports only decompression of brotli buffers.*

The ios libraries are compiled as universal and bitcode enabled. That means that they will support 32 and 64 bit builds.
tvOS is supported.

(**non-bitcode enabled iOS plugins are provided. If you are creating a non bitcode enabled project please use these provided plugins**!)

OSX bundle is compiled now as 64-bit only, since Apple store requires it. An x86_64 + Silicon version is included as a zip. If you need silicon support, unzip and replace the .bundle.
The Windows and Linux libraries are compiled for x86 and x86_64 build modes.
The Android lib is compiled for armeabi-v7a, x86, x86_64 and arm64-v8a.

FEATURES:

Fast brotli compression and decompression with a clean and simple interface. Very easy to use.

- compress a file into a brotli file format.

- decompress a brotli file.

- compress a buffer into the brotli format.

- decompress a brotli compressed buffer into a buffer.

(The plugin provides a solution to store the uncompressed size of a compressed buffer in its footer.)

- Linux, iOS, Android, MacOSX can treat buffers as files. That means if you have a file in www.bytes or in a native buffer 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.

!!! If you want to use only the brotli plugin, please delete all the other plugins in their respective folders or use the single packages from the _plugin_packages folder!!!.

INSTRUCTIONS:

If you want to run a small example, compile and run the testScene.
It will download a small tif file and it will perform all the functions the lib provides.

See the brotli.cs file for more comments and error codes.

In your project include in the Plugins folder the plugins you want to use and the brotli.cs file and call the appropriate functions as described below and shown in the demo scene.

FUNCTIONS:

```
int compressFile(string inFile, string outFile,  ulong[] proc, int quality = 9, int lgwin = 19, int lgblock = 0, int mode = 0);
```

Compress a file to brotli format.

Full paths to the files should be provided.

InFile : The input file
outFile: The output file
proc : A single item ulong array to provide progress of compression

quality: (0 - 11) quality of compression (0 = faster/bigger - 11 = slower/smaller).

lgwin : Base 2 logarithm of the sliding window size. Range is 10 to 24. (10 - 24) memory used for compression (higher numbers use more ram)

lgblock: 0 for auto or 16-24. Base 2 logarithm of the maximum input block size. Range is 16 to 24. If set to 0, the value will be set based on the quality.

mode : (0 - 2) 0 = default, 1 = utf8 text, 2 = woff 2.0 font

error codes: 1 : OK
-1 : compression failed
-2 : not enough memory
-3 : could not close in file
-4 : could not close out file
-5 : no input file found

```
int decompressFile(string inFile, string outFile, ulong[] proc, object fileBuffer = null);
```

Decompress a brotli file.

Full paths to the files should be provided.

```
InFile          : The input file
outFile         : The output file
proc            : A single item ulong array to provide progress of decompression
fileBuffer      : A buffer that holds a brotli file. When assigned the function will decompress from
                  this buffer and will ignore the filePath. (iOS, Android, MacOSX, Linux)
                : It can be a byte[] buffer or a native IntPtr buffer (downloaded using the helper
                  function: downloadLZ4FileNative / see demo script)
                : When an IntPtr is used as the input buffer, the size of it must be passed to the
                  function as a string with the inFile parameter!
returns         : 1 on success.

error codes     :  1  : OK
                  -1  : failed to write output
                  -2  : corrupt input
                  -3  : could not close in file
                  -4  : could not close out file
                  -5  : no input file found
```

```
int getDecodedSize(byte[] inBuffer);
```

Get the uncompressed size of a brotli buffer. This will work only on small buffers with one metablock. Otherwise use the includeSize/hasFooter flags with the buffer functions.

```
inBuffer:       the input buffer that stores a brotli compressed buffer.
```

```
bool compressBuffer(byte[] inBuffer, ref byte[] outBuffer, ulong[] proc, bool includeSize = false,
int quality = 9, int lgwin = 19, int lgblock = 0, int mode = 0);
```

Compress a byte buffer in brotli format.
Returns true on success.

```
inBuffer        : the uncompressed buffer.
outBuffer       : a referenced buffer that will store the compressed data. (it should be large enough
                  to store it.)
proc            : A single item referenced ulong array to provide progress of compression

includeSize     : include the uncompressed size of the buffer in the resulted compressed one because
                          brotli does not support it for larger then 1 metablock.

quality         : (0 - 11) quality of compression (0 = faster/bigger - 11 = slower/smaller).

lgwin           : (10 - 24) memory used for compression (higher numbers use more ram). Base 2
                          logarithm of the sliding window size. Range is 10 to 24.

lgblock         : 0 for auto or 16-24. Base 2 logarithm of the maximum input block size. Range is 16
                          to 24. If set to 0, the value will be set based on the quality.

Mode            : (0  -  2) 0 = default, 1 = utf8 text, 2 = woff 2.0 font
```

```
byte[] compressBuffer(byte[] inBuffer,  int[] proc, bool includeSize = false, int quality = 9, int
lgwin = 19, int lgblock = 0, int mode = 0);
```

Same as above only this function returns a new created buffer with the compressed data.

```
int compressBuffer(byte[] inBuffer, byte[] outBuffer, int[] proc, bool includeSize = false, int
quality = 9, int lgwin = 19, int lgblock = 0, int mode = 0);
```

*Same as **bool compressBuffer**, only this time the compressed buffer is written in a **fixed** size buffer.*
The fixed size buffer should be larger then the compressed size returned.

Returns the compressed size in bytes.

```
bool decompressBuffer(byte[] inBuffer, ref byte[] outBuffer, bool useFooter = false, int
unCompressedSize = 0);
```

Decompress a brotli compressed buffer to a referenced buffer.
Returns true on success.

inBuffer   : the brotli compressed buffer
outBuffer   : a referenced buffer that will be resized to store the uncompressed data.
useFooter   : if the input Buffer has the uncompressed size info.
UnCompressedSize: if unCompressedSize is > 0 then this is the uncompressed size that will be used.
Useful when decompressing brotli buffers created from servers.

```
byte[] decompressBuffer(byte[] inBuffer, bool useFooter = false, int unCompressedSize = 0);
```

Same as above only this time the uncompressed data is returned in a new created buffer.

```
int decompressBuffer(byte[] inBuffer, byte[] outBuffer, bool useFooter = false, int unCompressedSize
= 0);
```

Same as above only the decompressed data will be stored in a fixed size outBuffer.
Make sure the fixed buffer is big enough to store the data.

Returns: uncompressed size in bytes.

```
[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.

Helper function

```
IEnumerator downloadBrFileNative(string url, Action<bool> downloadDone, Action<IntPtr> pointer
= null, Action<int> fileSize = null);
```

A Coroutine to dowload a file to a native/unmaged memory buffer.
You can call it for an IntPtr.
See brotlitest.cs for usage example.

This function can only be called for one file at a time. Don't use it to call multiple files at once.

This is useful to avoid memory spikes when downloading large files and intend to decompress from memory. With the old method, a copy of the downloaded file to memory would be produced by pinning the buffer to memory.
Now with this method, it is downloaded to memory and can be manipulated with no memory spikes.

In any case, if you don't need the created in-Memory file, you should use the brotli.brReleaseBuffer function to free the memory!

Parameters:

url:	The url of the file you want to download to a native memory buffer.
downloadDone:	Informs a bool that the download of the file to memory is done.
pointer:	An IntPtr for a native memory buffer
fileSize:	The size of the downloaded file will be returned here.

SUPPORT:

For any questions, problems and suggestions please use this email address:
elias_t@yahoo.com

forum: http://forum.unity3d.com/threads/7zip-lzma-and-zip-native-multiplatform-