xbcompress Sample

*This sample is compatible with the Microsoft Game Development Kit (November 2020)*

# Description

This samples demonstrates the [Compression API](https://docs.microsoft.com/en-us/windows/win32/cmpapi/-compression-portal) introduced with Windows 8 which is supported for all Gaming.\*.x64 platforms.

Running the tool without any parameters shows the help screen, as follows:

Text

Description automatically generated

# Building the sample

As a simple command-line tool, you can build directly using the *Gaming Command Prompt*:

cl /EHsc /D\_WIN32\_WINNT=0x0A00 /Ox /MT compresstool.cpp /Fexbcompress.exe xgameplatform.lib

You can use CMake 3.15 or later:

cmake -B out .

cmake –build out

There are CMake Presets as well (introduced in CMake 3.19):

cmake --list-presets

cmake --preset=x64-Debug

cmake --build out\build\x64-Debug

Or you can open the CMakeLists.txt from the VS IDE (VS 2019 16.3 or later is required for CMake 3.15 integration).

* We build with the static Visual C++ Runtime to make the tool trivial to “xbcp deploy” to the console. Generally, we recommend using /MD for the DLL-based runtime for titles.
* The APIs used by this command-line tool are in onecore\_apiset.lib, xgameplatform.lib, and WindowsApp.lib. You can use the onecore\_apiset.lib umbrella lib safely for both PC and Xbox in this case (which is how the CMake is configured). Again, we recommend using xgameplatform.lib instead of any other umbrella library or kernel32.lib for titles.
* You could build this tool with \_WIN32\_WINNT=0x0602 (Windows 8) or \_WIN32\_WINNT=0x0603 (Windows 8.1) linking with cabinet.lib instead of onecore\_apiset.lib. Windows 7 or earlier does not support the Compression API.

# Usage

*This tool is intended for use for development scenarios where a “quick & dirty” CPU-based compression solution with a minimum of dependencies is called for: test automation, samples, demos, rapid-prototypes, etc.* ***For retail content scenarios, there are many other options including DirectStorage, BCPack, 3rd party libraries, and traditional ‘file-system-in-a-file’ solutions which are much more appropriate.***

This sample is a simple command-line tool that is compatible with Windows 10 Host PCs, Xbox System OS, and Xbox Game OS. You can use it to compress or decompress files.

xbcompress.exe mylargefile.bin

-or-

xbcp /x/title xbcompress.exe xd:\

xbrun /x/title /O d:\xbcompress.exe d:\mylargefile.bin

This results in ‘mylargefile.bi\_’ being written to the current directory or D:\ directory. By default this file is compressed using LZMS compression.

To expand the file, use the **/u** switch

xbcompress /u mylargefile.bi\_

-or-

xbrun /x/title /O d:\xbcompress.exe /u d:\mylargefile.bi\_

This will result in the ‘mylargefile.bin’ being written to the current directory or D:\.

The LZMS compression scheme is considered a good choice for files over 2 Mbytes in size. If you want a slightly faster compress speed with a slightly less compact size, you can use the **/z** switch to compress with MSZIP instead.

# Implementation

This sample takes its inspiration from the classic MS-DOS utilities COMPRESS.EXE and EXPAND.EXE. The ‘\_’ files produced by this tool are not compatible or recognized by the OS tool EXPAND.EXE. The compressed file always ends with ‘\_’. If the file extension is 3 or more character long, the last character is replaced by ‘\_’. Otherwise ‘.\_’ is appended as an extension.

To keep the code extremely simple, the tool uses the Compression API ‘buffer’ mode. The API manages breaking up the data into blocks and encodes the metadata needed to decompress in the compressed data block.

Compressed files start with the following simple header:

|  |  |  |
| --- | --- | --- |
| File offset | Field length | Description |
| 0 | 8 | Magic byte sequence to uniquely identify file format.  0x41, 0x46, 0x43, 0x57, 0x47, 0x50, 0x53, 0x4d |
| 9 | 1 | Compression mode.  Only supported modes currently are:   * COMPRESS\_ALGORITHM\_LZMS (5) * COMPRESS\_ALGORITHM\_MSZIP (2) |
| 10 | 1 | File format version.  Currently 0x41 ('A') |
| 11 | 2 | Last character (UTF-16LE) that was changed to '\_' when the compressed name was determined. This value is 0 if '.\_' was added instead. |
| 13 | 4 | Size in bytes of the original uncompressed data block.  *To keep the code simple, this file format only supports up to 4 GB file sizes.* |

And example of runtime code to decompress a file produced by XBCOMPRESS.EXE can be found in ATGTK\ReadCompressedData.h / .cpp.

# Update history

|  |  |
| --- | --- |
| Release | Notes |
| April 2021 | Initial release |
| January 2022 | CMake cleanup and added presets file |