Framework

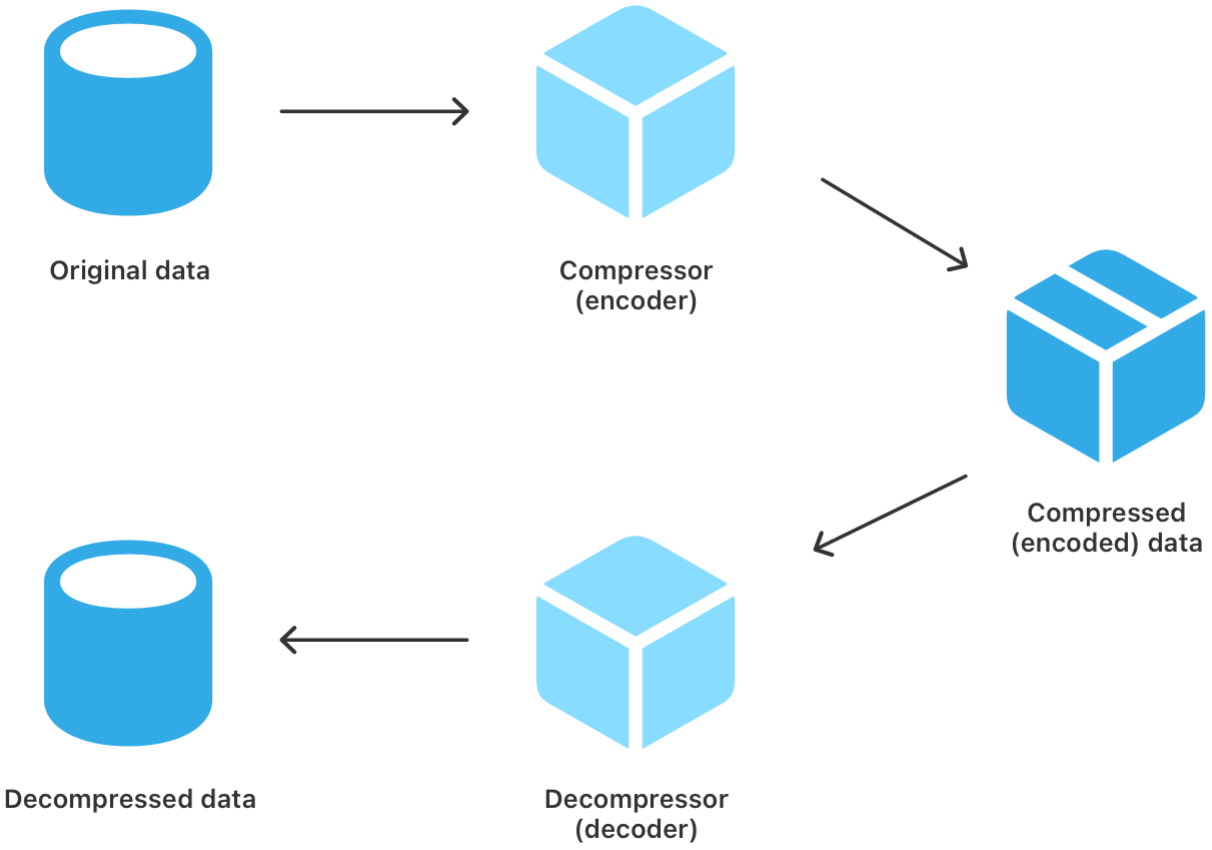
Compression

Leverage common compression algorithms for lossless data compression.

iOS 9.0+ | iPadOS 9.0+ | Mac Catalyst 13.1+ | macOS 10.11+ | tvOS 9.0+ | visionOS 1.0+ | watchOS 2.0+

Overview

The Compression framework enables your app to provide lossless compression when saving or sharing files and data. Compression is a process in which you compress (encode) and decompress (decode) data. For example, a text editor may save its files in a compressed format, and automatically decompress the saved file when the user opens it.



The framework offers two methods of compression:

- *Buffer compression* uses a single-step method for compressing files, making it perfect for use with uncompressed files under 8 MB, or compressed files under 1 MB.
- *Stream compression* uses multiple steps for compressing files, making it ideal for compressing larger files or streamed data, such as an incoming audio signal or downloading files.

To use buffer compression, you compress or decompress the input data with one call to the corresponding function. To learn more about buffer compression, including a walk-through of the code used to encode and decode a string, see Compressing and decompressing data with buffer compression.

To use stream compression, you call the compression or decompression function repeatedly to compress or decompress data from a source buffer to a destination buffer. Between calls, the compressor or decompressor moves processed data out of the source buffer and loads new data into the destination buffer. To learn more about stream compression, see the sample code project Compressing and decompressing files with stream compression.

Topics

Objects that simplify multiple-step compression

Simplify encoding and decoding streams of data using Compression classes for Swift.

Compressing and decompressing data with input and output filters

Compress and decompress streamed or from-memory data, using input and output filters.

{} Compressing and decompressing files with stream compression

Perform compression for all files and decompression for files with supported extension types.

`class InputFilter`

An encoder-decoder that reads input data from a stream.

`class OutputFilter`

An encoder-decoder that writes output data to a stream.

`enum Algorithm`

Algorithms used for compression or decompression.

`enum FilterError`

Errors that occur during compression.

`enum FilterOperation`

Operations that define whether input and output filters compress or decompress data.

Multiple-step compression

Stream compression functions compress or decompress sequential blocks of data.

`struct compression_stream`

A structure representing a compression stream.

`func compression_stream_init(UnsafeMutablePointer<compression_stream>, compression_stream_operation, compression_algorithm) -> compression_status`

Initializes a compression stream for either compression or decompression.

`func compression_stream_process(UnsafeMutablePointer<compression_stream>, Int32) -> compression_status`

Performs compression or decompression using an initialized compression stream structure.

`func compression_stream_destroy(UnsafeMutablePointer<compression_stream>) -> compression_status`

Frees any memory allocated by stream initialization function.

`struct compression_status`

A set of values used to represent the status of stream compression.

`struct compression_stream_flags`

A set of values used to represent stream compression flags.

`struct compression_stream_operation`

A set of values used to represent a stream compression operation.

`struct compression_algorithm`

A structure for values that represent compression algorithms.

Single-step compression

Buffer compression functions compress or decompress a block of data stored contiguously in memory.

Compressing and decompressing data with buffer compression

Compress a string, write it to the file system, and decompress the same file using buffer compression.

```
func compression_encode_scratch_buffer_size(compression_algorithm) -> Int
```

Returns the required compression scratch buffer size for the selected algorithm.

```
func compression_encode_buffer(UnsafeMutablePointer<UInt8>, Int, UnsafePointer<UInt8>, Int, UnsafeMutableRawPointer?, compression_algorithm) -> Int
```

Compresses the contents of a source buffer into a destination buffer.

```
func compression_decode_scratch_buffer_size(compression_algorithm) -> Int
```

Returns the required decompression scratch buffer size for the selected algorithm.

```
func compression_decode_buffer(UnsafeMutablePointer<UInt8>, Int, UnsafePointer<UInt8>, Int, UnsafeMutableRawPointer?, compression_algorithm) -> Int
```

Decompresses the contents of a source buffer into a destination buffer.

```
struct compression_algorithm
```

A structure for values that represent compression algorithms.