Function

compression_stream_process(_:_:)

Performs compression or decompression using an initialized compression stream structure.

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

```
func compression_stream_process(
    _ stream: UnsafeMutablePointer<compression_stream>,
    _ flags: Int32
) -> compression_status
```

Parameters

`stream`
A pointer to an allocated and fully initialized `compression_stream` structure.

`flags`
A constant of type `compression_stream_flags`; this should be `COMPRESSION_STREAM_FINALIZE` if there is no further input data, or `0` otherwise.

Return Value

A value of type `compression_status`, interpreted as follows:

- `COMPRESSION_STATUS_OK` means that processing was successful, but the stream may produce more output. Call the function again with updated parameters.
- `COMPRESSION_STATUS_END` means that processing was successful, and the stream will produce no more output (this only occurs if `flags` is set to `COMPRESSION_STREAM_FINALIZE`).

- `COMPRESSION_STATUS_ERROR` means an error occurred.

Discussion

Each time `compression_stream_process(_:_:)` is called successfully, the function consumes data from the source buffer and writes data into the destination buffer, until it reaches the end of one of the buffers and returns either `COMPRESSION_STATUS_OK` or `COMPRESSION_STATUS_END`.

After a successful call, the function updates the buffer parameters in the stream object: the function increments `src_ptr` (and decrements `src_size`) by the number of input bytes consumed. Likewise, the function incremets `dst_ptr` (and decrements `dst_size`) by the number of output bytes produced. The sum (`src_ptr + src_size`) remains unchanged, and so does (`dst_ptr + dst_size`). At this point, either `src_size` or `dst_size` will be 0, indicating that the source buffer is empty or the destination buffer is full.

If the source buffer is empty, you can refill it with more data and adjust the parameters, or point to a different buffer for the next call. If you don't provide any more input data, set `flags` to `COMPRESSION_STREAM_FINALIZE` and call again.

If the destination buffer is full and the return value is not `COMPRESSION_STATUS_END`, there may still be input available for processing. To let this happen, you might grow the buffer, move the pointer back to reuse the buffer, or point to a new destination buffer, and then call again.

See Also

Multiple-step compression

`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_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.