API Collection

VTCompressionSession

An object that compresses video data.

Overview

A compression session supports the compression of a sequence of video frames. Here's the workflow:

1. Create a compression session using `VTCompressionSessionCreate(allocator:width:height:codecType:encoderSpecification:imageBufferAttributes:compressedDataAllocator:outputCallback:refcon:compressionSessionOut:)`.
2. Optionally, configure the session with your desired Compression Properties by calling `VTSessionSetProperty(_:key:value:)` or `VTSessionSetProperties(_:propertyDictionary:)`.
3. Encode video frames using `VTCompressionSessionEncodeFrame(_:imageBuffer:presentationTimeStamp:duration:frameProperties:sourceFrameRefcon:infoFlagsOut:)` and receive the compressed video frames in the session's `VTCompressionOutputCallback`.
4. To force the completion of some or all pending frames, call `VTCompressionSessionCompleteFrames(_:untilPresentationTimeStamp:)`.
5. When you finish with the compression session, call `VTCompressionSessionInvalidate(_:)` to invalidate it and `CFRelease` to free its memory.

Topics

Creating a Session

```
func VTCompressionSessionCreate(allocator: CFAllocator?, width: Int32, height: Int32, codecType: CMVideoCodecType, encoderSpecification: CFDictionary?, imageBufferAttributes: CFDictionary?, compressedDataAllocator: CFAllocator?, outputCallback: VTCompressionOutputCallback?, refcon: UnsafeMutableRawPointer?, compressionSessionOut: UnsafeMutablePointer<VTCompressionSession?>) -> OSStatus
```

Creates an object that compresses video frames.

Configuring a Session

Compression Properties

Properties that you use to configure a compression session.

Encoding Frames

```
func VTCompressionSessionGetPixelBufferPool(VTCompressionSession) -> CVPixelBufferPool?
```

Returns a pool that provides ideal source pixel buffers for a compression session.

```
func VTCompressionSessionPrepareToEncodeFrames(VTCompressionSession) -> OSStatus
```

Enables the encoder to perform any necessary resource allocation before the encoder begins encoding frames (optional).

```
func VTCompressionSessionEncodeFrame(VTCompressionSession, imageBuffer: CVImageBuffer, presentationTimeStamp: CMTime, duration: CMTime, frameProperties: CFDictionary?, sourceFrameRefcon: UnsafeMutableRawPointer?, infoFlagsOut: UnsafeMutablePointer<VTEncodeInfoFlags>?) -> OSStatus
```

Presents frames to the compression session.

```
func VTCompressionSessionEncodeFrame(VTCompressionSession, imageBuffer: CVImageBuffer, presentationTimeStamp: CMTime, duration: CMTime, frameProperties: CFDictionary?, infoFlagsOut: UnsafeMutablePointer<VTEncodeInfoFlags>?, outputHandler: VTCompressionOutputHandler) -> OSStatus
```

Presents frames to the compression session and invokes the output callback when compression is complete.

```
func VTCompressionSessionCompleteFrames(VTCompressionSession, untilPresentationTimeStamp: CMTime) -> OSStatus
```

Forces the compression session to complete the encoding of frames.

Encoding Multi-Image Frames

```
func VTIsStereoMVHEVCEncodeSupported() -> Bool
```

Returns a Boolean value that indicates whether the system supports MV-HEVC encoding.

```
func VTCompressionSessionEncodeMultiImageFrame(VTCompressionSession, taggedBuffers: [CMTaggedBuffer], presentationTimeStamp: CMTime, duration: CMTime, frameProperties: CFDictionary?, infoFlagsOut: UnsafeMutablePointer<VTEncodeInfoFlags>?, outputHandler: VTCompressionOutputHandler) -> OSStatus
```

Passes a multi-image frame to a compression session for encoding and provides a callback to handle the output.

Performing Multiple Passes

```
func VTCompressionSessionBeginPass(VTCompressionSession, flags: VTCompressionSessionOptionFlags, UnsafeMutablePointer<UInt32>?) -> OSStatus
```

Marks the start of a specific compression pass.

```
func VTCompressionSessionEndPass(VTCompressionSession, furtherPassesRequestedOut: UnsafeMutablePointer<DarwinBoolean>?, UnsafeMutablePointer<UInt32>?) -> OSStatus
```

Marks the end of a compression pass.

```
func VTCompressionSessionGetTimeRangesForNextPass(VTCompressionSession, timeRangeCountOut: UnsafeMutablePointer<CMItemCount>, timeRangeArrayOut: UnsafeMutablePointer<UnsafePointer<CMTimeRange>?>) -> OSStatus
```

Retrieves the time ranges for the next pass.

Invalidating a Session

```
func VTCompressionSessionInvalidate(VTCompressionSession)
```

Tears down a compression session.

Accessing the Type Identifier

```
func VTCompressionSessionGetTypeID() -> CFTypeID
```

Retrieves the Core Foundation type identifier for the compression session.

Data Types

`class` `VTCompressionSession`

A reference to a VideoToolbox compression session.

`struct` `VTEncodeInfoFlags`

Flags that indicate encoder state.

See Also

Compression

{} Encoding video for low-latency conferencing

Configure a compression session to optimize encoding for video-conferencing apps.

{} Encoding video for live streaming

Configure a compression session to encode video for live streaming.

{} Encoding video for offline transcoding

Configure a compression session to transcode video in offline workflows.

VTDecompressionSession

An object that decompresses video data.

VTFrameSilo

An object that stores sample buffers from a multipass encoding session.

VTMultiPassStorage

An object that stores video encoding metadata from a multipass encoding session.