Framework

Video Toolbox

Work directly with hardware-accelerated video encoding and decoding capabilities.

iOS 6.0+ | iPadOS 6.0+ | Mac Catalyst 13.0+ | macOS 10.8+ | tvOS 10.2+ | visionOS 1.0+

Overview

VideoToolbox is a low-level framework that provides direct access to hardware encoders and decoders. It provides services for video compression and decompression, and for conversion between raster image formats stored in CoreVideo pixel buffers. These services are provided in the form of session objects (compression, decompression, and pixel transfer), which are vended as Core Foundation (CF) types. Apps that don’t need direct access to hardware encoders and decoders shouldn’t need to use VideoToolbox directly.

Topics

Frame Processing

Frame processing

An interface for accessing a range of different video-processing features.

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.

VTCompressionSession

An object that compresses video data.

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.

Transformation

VTPixelTransferSession

An object converts video data from source pixel buffers to destination pixel buffers.

VTPixelRotationSession

An object that rotates source pixel buffers to destination pixel buffers.

RAW Processing

`class VTRAWProcessingSession`

An object that processes frames in camera native formats such as RAW or Bayer.

Media Extension

`struct VTExtensionPropertiesKey`

A key in a Media Extension extension properties dictionary.

HDR Metadata

`class VTHDRPerFrameMetadataGenerationSession`

An object that generates per-frame HDR metadata.

Codec Support

```
func VTIsHardwareDecodeSupported(CMVideoCodecType) -> Bool
```

Returns a Boolean value that indicates whether the current system supports hardware decode for the specified codec.

```
func VTRegisterProfessionalVideoWorkflowVideoEncoders()
```

Loads encoders appropriate for the client's professional video workflows.

```
func VTRegisterProfessionalVideoWorkflowVideoDecoders()
```

Loads decoders appropriate for the client's professional video workflows.

```
func VTRegisterSupplementalVideoDecoderIfAvailable(CMVideoCodecType)
```

Registers a video decoder for the specified codec type, if one exists on the current system.

```
func VTCopySupportedPropertyDictionaryForEncoder(width: Int32, height:
Int32, codecType: CMVideoCodecType, encoderSpecification: CFDictionary
?, encoderIDOut: UnsafeMutablePointer<CFString?>?, supportedProperties
Out: UnsafeMutablePointer<CFDictionary?>?) -> OSStatus
```

Builds a list of supported properties and encoder ID for an encoder.

```
func VTCopyVideoEncoderList(CFDictionary?, UnsafeMutablePointer<CFArray
?>) -> OSStatus
```

Builds a list of available video encoders.

Video Encoder List Keys

Dictionary key constants to use to retrieve video encoder information.

Utilities

```
func VTCreateCGImageFromCVPixelBuffer(CVPixelBuffer, options:
CFDictionary?, imageOut: UnsafeMutablePointer<CGImage?>) -> OSStatus
```

Creates a Core Graphics bitmap image or image mask using the provided pixel buffer.

Data Types

VTSession

An abstract object that provides the common interface to configure VideoToolbox session objects.

```
struct VTInt32Point
```

A structure that represents a 32-bit integer point value.

`struct VTInt32Size`

A structure that represents a 32-bit integer size value.

`var VT_SUPPORT_COLORSYNC_PIXEL_TRANSFER: Bool`

Errors

Error Code Constants

Constants for Video Toolbox operation error codes.

Reference

VideoToolbox Reference

Classes

`class VTLowLatencyFrameInterpolationConfiguration`

Configuration that you use to program Video Toolbox frame processor for low-latency frame interpolation.

`class VTLowLatencyFrameInterpolationParameters`

An object that contains both input and output parameters that the low-latency frame interpolation processor needs.

`class VTLowLatencySuperResolutionScalerConfiguration`

An object you use to configure frame processor for low-latency super-resolution scaler processing.

`class VTLowLatencySuperResolutionScalerParameters`

An object that contains both input and output parameters that the low-latency super-resolution scaler frame processor needs.

`class VTMotionEstimationSession`

`class VTSuperResolutionScalerConfiguration`

Configuration that you use to set up the super-resolution processor.

`class VTSuperResolutionScalerParameters`

An object that contains both input and output parameters that the super-resolution processor needs to run on a frame.

`class VTTemporalNoiseFilterConfiguration`

A configuration object to initiate a frame processor and use temporal noise-filter processor.

`class VTTemporalNoiseFilterParameters`

Encapsulates the frame-level parameters necessary for processing a source frame using temporal noise-filter processor.

Variables

`let kVTCameraCalibrationExtrinsicOriginSource_StereoCameraSystemBaseline: CFString`

`let kVTCameraCalibrationLensAlgorithmKind_ParametricLens: CFString`

`let kVTCameraCalibrationLensDomain_Color: CFString`

`let kVTCameraCalibrationLensRole_Left: CFString`

`let kVTCameraCalibrationLensRole_Mono: CFString`

`let kVTCameraCalibrationLensRole_Right: CFString`

`let kVTCompressionPreset_Balanced: CFString`

`let kVTCompressionPreset_HighQuality: CFString`

`let kVTCompressionPreset_HighSpeed: CFString`

`let kVTCompressionPreset_VideoConferencing: CFString`

`let kVTCompressionPropertyCameraCalibrationKey_ExtrinsicOrientationQuaternion: CFString`

`let kVTCompressionPropertyCameraCalibrationKey_ExtrinsicOriginSource: CFString`

`let kVTCompressionPropertyCameraCalibrationKey_IntrinsicMatrix: CFString`

`let kVTCompressionPropertyCameraCalibrationKey_IntrinsicMatrixProjectionOffset: CFString`

`let kVTCompressionPropertyCameraCalibrationKey_IntrinsicMatrixReferenceDimensions: CFString`

`let kVTCompressionPropertyCameraCalibrationKey_LensAlgorithmKind: CFString`

The following keys are required in each kVTCompressionPropertyKey_CameraCalibrationDataLensCollection dictionary.

```
let kVTCompressionPropertyCameraCalibrationKey_LensDistortions:
CFString
```

```
let kVTCompressionPropertyCameraCalibrationKey_LensDomain: CFString
```

```
let kVTCompressionPropertyCameraCalibrationKey_LensFrameAdjustments
PolynomialX: CFString
```

```
let kVTCompressionPropertyCameraCalibrationKey_LensFrameAdjustments
PolynomialY: CFString
```

```
let kVTCompressionPropertyCameraCalibrationKey_LensIdentifier: CFString
```

```
let kVTCompressionPropertyCameraCalibrationKey_LensRole: CFString
```

```
let kVTCompressionPropertyCameraCalibrationKey_RadialAngleLimit:
CFString
```

```
let kVTCompressionPropertyKey_CameraCalibrationDataLensCollection:
CFString
```

```
let kVTCompressionPropertyKey_SupportedPresetDictionaries: CFString
```

```
let kVTCompressionPropertyKey_VBVBufferDuration: CFString
```

```
let kVTCompressionPropertyKey_VBVInitialDelayPercentage: CFString
```

```
let kVTCompressionPropertyKey_VBVMaxBitRate: CFString
```

```
let kVTCompressionPropertyKey_VariableBitRate: CFString
```

```
let kVTDecodeFrameOptionKey_ContentAnalyzerCropRectangle: CFString
```

```
let kVTDecodeFrameOptionKey_ContentAnalyzerRotation: CFString
```

```
let kVTHDRMetadataInsertionMode_RequestSDRRangePreservation: CFString
```

```
let kVTHeroEye_Left: CFString
```

```
let kVTHeroEye_Right: CFString
```

```
let kVTMotionEstimationSessionCreationOption_Label: CFString!
```

A label you use to log and track resources.

```
let kVTMotionEstimationSessionCreationOption_MotionVectorSize: CFString
!
```

The size of the search blocks that motion estimation session uses.

```
let kVTMotionEstimationSessionCreationOption_UseMultiPassSearch:
CFString!
```

An option to use for higher quality motion estimation.

```
let kVTProjectionKind_Equirectangular: CFString
```

```
let kVTProjectionKind_HalfEquirectangular: CFString
```

```
let kVTProjectionKind_ParametricImmersive: CFString
```

```
let kVTProjectionKind_Rectilinear: CFString
```

```
let kVTRAWProcessingPropertyKey_MetadataForSidecarFile: CFString!
```

```
var kVTVideoEncoderAutoWhiteBalanceNotLockedErr: OSStatus
```

```
let kVTViewPackingKind_OverUnder: CFString
```

```
let kVTViewPackingKind_SideBySide: CFString
```

Functions

```
func VTDecompressionSessionDecodeFrame(VTDecompressionSession, sample
Buffer: CMSampleBuffer, flags: VTDecodeFrameFlags, frameOptions:
CFDictionary?, frameRefcon: UnsafeMutableRawPointer?, infoFlagsOut:
UnsafeMutablePointer<VTDecodeInfoFlags>?) -> OSStatus
```

```
func VTDecompressionSessionDecodeFrame(VTDecompressionSession, sample
Buffer: CMSampleBuffer, flags: VTDecodeFrameFlags, frameOptions:
CFDictionary?, infoFlagsOut: UnsafeMutablePointer<VTDecodeInfoFlags>?,
outputHandler: VTDecompressionOutputHandler) -> OSStatus
```

Type Aliases

```
typealias VTMotionEstimationOutputHandler
```

A block invoked by motion-estimation session when frame processing is complete.