```
import CoreAudio
import CoreAudioTypes
import CoreFoundation
import CoreGraphics
import CoreMedia CMAttachment
import CoreMedia CMAudioClock
import CoreMedia CMAudioDeviceClock
import CoreMedia CMBase
import CoreMedia CMBlockBuffer
import CoreMedia CMBufferQueue
import CoreMedia CMFormatDescription
import CoreMedia CMFormatDescriptionBridge
import CoreMedia CMMemoryPool
import CoreMedia CMMetadata
import CoreMedia CMSampleBuffer
import CoreMedia CMSimpleQueue
import CoreMedia CMSync
import CoreMedia CMTag
import CoreMedia CMTagCollection
import CoreMedia CMTaggedBufferGroup
import CoreMedia CMTextMarkup
import CoreMedia CMTime
import CoreMedia CMTimeRange
import CoreVideo
import Darwin
import Dispatch
import Foundation
import _Concurrency
import _StringProcessing
import _SwiftConcurrencyShims

@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
public struct CMAttachmentBearerAttachments

    /// Type to specify attachment.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public enum Value

        case shouldNotPropagate Any

        case shouldPropagate Any

        /// The value of the attachment
        public var value  Any   get

        /// The mode of the attachment.
        public var mode  CMAttachmentBearerAttachments Mode
```

```
get

    ///  The attachment modes are the same as those defined in
CMAttachment.h.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public enum Mode   CMAttachmentMode  Sendable

        case shouldNotPropagate

        case shouldPropagate

        ///  Creates a new instance with the specified raw value.
        ///
        ///  If there is no value of the type that corresponds with the specified
raw
        ///  value, this initializer returns `nil`. For example:
        ///
        ///      enum PaperSize: String {
        ///          case A4, A5, Letter, Legal
        ///      }
        ///
        ///      print(PaperSize(rawValue: "Legal"))
        ///      // Prints "Optional("PaperSize.Legal")"
        ///
        ///      print(PaperSize(rawValue: "Tabloid"))
        ///      // Prints "nil"
        ///
        /// - Parameter rawValue:  The raw value to use for the new
instance.
        public init              CMAttachmentMode

        ///  The raw type that can be used to represent all values of the
conforming
        ///  type.
        ///
        ///  Every distinct value of the conforming type has a corresponding
unique
        ///  value of the `RawValue` type, but there may be values of the
`RawValue`
        ///  type that don't have a corresponding value of the conforming type.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias RawValue   CMAttachmentMode

        ///  The corresponding value of the raw type.
        ///
        ///  A new instance initialized with `rawValue` will be equivalent to this
        ///  instance. For example:
```

```
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter, Legal
        ///     }
        ///
        ///     let selectedSize = PaperSize.Letter
        ///     print(selectedSize.rawValue)
        ///     // Prints "Letter"
        ///
        ///     print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue)!)
        ///     // Prints "true"
        public var rawValue  CMAttachmentMode   get


    /// Accesses the attachment associated with the given key for reading and
    /// writing.
    ///
    /// You can attach any object to a `CMAttachmentBearerProtocol`
object to
    /// store additional information.
    ///
    /// If the key doesn't exist, the attachment will be added.
    ///
    /// If the key does exist, the existing attachment will be replaced.
    ///
    /// If you assign `nil` as the value for the given key, the attachment bearer
    /// removes that key and its associated value.
    ///
    /// - Parameter key: Key identifying the desired attachment.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public subscript      String
CMAttachmentBearerAttachments Value

    /// Removes all attachments of a `CMAttachmentBearerProtocol`
object.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func removeAll

    /// Dictionary of non propagated attachments.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var nonPropagated   String   Any    get

    /// Dictionary of propagated attachments.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var propagated   String   Any    get
    /// Sets a set of attachments for a `CMAttachmentBearerProtocol`
object.
    ///
    /// `attachments.merge(_:mode:)` is a convenience call that in turn
calls
    /// `attachments[key] = mode(value)` for each key and value in the
given
    /// dictionary
    ///
    /// - Parameters:
    ///   - attachments: Attachments to attach and their keys.
    ///   - mode: The mode of the attachments.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func merge _                 String   Any
CMAttachmentBearerAttachments Mode


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMAttachmentBearerAttachments

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public subscript      CMSampleBuffer AttachmentKey
CMAttachmentBearerAttachments Value


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMAttachmentBearerAttachments Mode   Equatable


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMAttachmentBearerAttachments Mode   Hashable


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMAttachmentBearerAttachments Mode
RawRepresentable


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
public protocol CMAttachmentBearerProtocol

    /// Access attachments.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0

    var attachments  CMAttachmentBearerAttachments   get

    ///  Copy all propagatable attachments from one buffer to another.
    ///
    /// - Parameter destination: `CMAttachmentBearerProtocol`
object to copy
    ///    attachments to.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0

    func propagateAttachments T                  T  where T
CMAttachmentBearerProtocol


///  Methods that operate on a range of a `CMBlockBuffer` uses
/// `CMBlockBufferProtocol`.
@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
public protocol CMBlockBufferProtocol

    ///  `CMBlockBuffer` instance to operate on.
    var owner  CMBlockBuffer   get

    ///  The position of the first element.
    var startIndex  Int   get

    ///  The "past the end" position.
    var endIndex  Int   get


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMBlockBufferProtocol

    ///  Creates a slice from a `ClosedRange`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public subscript         ClosedRange Int
CMBlockBuffer Slice   get

    ///  Creates a slice from a `Range`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public subscript         Range Int
CMBlockBuffer Slice   get

    ///  Creates a slice from a `PartialRangeUpTo`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
```

```
visionOS 1.0
    public subscript         PartialRangeUpTo Int
CMBlockBuffer Slice    get

    /// Creates a slice from a `PartialRangeThrough`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public subscript         PartialRangeThrough Int
CMBlockBuffer Slice    get

    /// Creates a slice from a `PartialRangeFrom`.
    ///
    /// The endIndex is set to the current "past the end" position.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public subscript         PartialRangeFrom Int
CMBlockBuffer Slice    get

    /// Creates a slice from an `UnboundedRange`.
    ///
    /// The endIndex is set to the current "past the end" position.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public subscript          UnboundedRange_
CMBlockBuffer Slice    get


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMBlockBufferProtocol

    /// Produces a `CMBlockBuffer` containing a contiguous copy of or
reference to
    /// the data specified by the parameters.
    ///
    /// Produces a `CMBlockBuffer` containing a contiguous copy of or
reference to
    /// the data specified by the parameters. The resulting new
`CMBlockBuffer`
    /// may contain an allocated copy of the data, or may contain a contiguous
    /// `CMBlockBuffer` reference.
    ///
    /// If `.alwaysCopyData` is set in the flags parameter, the resulting
    /// `CMBlockBuffer` will contain an allocated copy of the data rather than
a
    /// reference to the source buffer.
    ///
    /// - Parameters:
    ///   - allocator:  Allocator to be used for allocating the memory block if
a
```

```
    ///       contiguous copy of the data is to be made.
    ///   - flags: Feature and control flags.
    /// - Returns: Newly-created `CMBlockBuffer` object with contiguous
memory
    ///   backing.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func makeContiguous            CFAllocator
                            CMBlockBuffer Flags       throws
   CMBlockBuffer

    /// Produces a `CMBlockBuffer` containing a contiguous copy of or
reference to
    /// the data specified by the parameters.
    ///
    /// Produces a `CMBlockBuffer` containing a contiguous copy of or
reference to
    /// the data specified by the parameters. The resulting new
`CMBlockBuffer`
    /// may contain an allocated copy of the data, or may contain a contiguous
    /// `CMBlockBuffer` reference.
    ///
    /// If `.alwaysCopyData` is set in the flags parameter, the resulting
    /// `CMBlockBuffer` will contain an allocated copy of the data rather than
a
    /// reference to source buffer.
    ///
    /// - Parameters:
    ///   - allocator: Allocator to be used for allocating the memory block if
a
    ///       contiguous copy of the data is to be made.
    ///   - deallocator: Deallocator to be used for deallocating the
memory block.
    ///   - flags: Feature and control flags.
    /// - Returns: Newly-created `CMBlockBuffer` object with contiguous
memory
    ///   backing.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func makeContiguous            @escaping
CMBlockBuffer CustomBlockAllocator               @escaping
CMBlockBuffer CustomBlockDeallocator
CMBlockBuffer Flags       throws    CMBlockBuffer


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMBlockBufferProtocol

    /// Accesses potentially noncontiguous data.
    ///
    /// Used for accessing potentially noncontiguous data, this routine will call
    /// `body` with a buffer pointer directly into the given `CMBlockBuffer` if
    /// possible, otherwise the data will be assembled and copied into a
    /// temporary block and `body` will be called with its buffer pointer.
    ///
    /// - Parameter body: A closure with an
`UnsafeRawBufferPointer` parameter
    ///     that points to contiguous storage for the block buffer.
    ///
    /// - Returns: The return value, if any, of the body closure parameter.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func withContiguousStorage R  _
 UnsafeRawBufferPointer  throws    R  throws    R

    /// Copies bytes to a `Data`.
    ///
    /// This function is used to copy bytes out of a `CMBlockBuffer`.
    /// It deals with the possibility of the desired range of data being
    /// noncontiguous.
    ///
    /// - Returns: `Data` containing the bytes requested.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func dataBytes   throws    Data

    /// Copies bytes into a provided memory area.
    ///
    /// This function is used to copy bytes out of a `CMBlockBuffer` into a
    /// provided piece of memory.
    ///
    /// It deals with the possibility of the desired range of data being
    /// noncontiguous.
    ///
    /// - Parameters:
    ///   - destination: Memory into which the data should be copied.
Must be
    ///       large enough to contain `dataLength` bytes.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func copyDataBytes
UnsafeMutableRawBufferPointer  throws

    /// Copies bytes from a given memory block, replacing bytes in the
underlying
    /// data blocks
    ///
    /// This function is used to replace bytes in a `CMBlockBuffer`'s memory
    /// blocks with those from a provided piece of memory.
    ///
    /// It deals with the possibility of the destination range of data being
    /// noncontiguous. `assureBlockMemory()` is called. If desired range is
    /// subsequently not accessible in the `CMBlockBuffer`, an error is
thrown and
    /// the contents of the `CMBlockBuffer` are untouched.
    ///
    /// - Parameters:
    ///   - sourceBytes: Memory block from which bytes are copied into the
    ///     `CMBlockBuffer`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func replaceDataBytes
UnsafeRawBufferPointer  throws

    /// Fills the `CMBlockBuffer` with a given byte value, replacing bytes in
the
    /// underlying data blocks
    ///
    /// This function is used to fill bytes in a `CMBlockBuffer`'s memory
blocks
    /// with a given byte value.
    ///
    /// It deals with the possibility of the destination range of data being
    /// noncontiguous. `assureBlockMemory()` is called. If desired range is
    /// subsequently not accessible in the `CMBlockBuffer`, an error is
thrown and
    /// the contents of the `CMBlockBuffer` are untouched.
    ///
    /// - Parameters:
    ///   - fillByte: The value with which to fill the specified data range.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func fillDataBytes               UInt8  throws


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMBlockBufferProtocol

    /// Obtains the total data length.
    ///
    /// Obtains the total data length. This total is the sum of the `dataLength`s
    /// of the `CMBlockBuffer`'s memory blocks and buffer references. Note
that
    /// the `dataLength`s are the _portions_ of those constituents that this
    /// `CMBlockBuffer` subscribes to. This `CMBlockBuffer` presents a
contiguous
    /// range of offsets from zero to its total `dataLength`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
```

```
visionOS 1.0
    public var dataLength  Int   get

    /// Determines whether the `CMBlockBuffer` is contiguous.
    ///
    /// If withUnsafeMutableBytes(atOffset:_:) were to be called with the
    /// same parameters, the returned buffer pointer would address the desired
    /// number of bytes.
    ///
    /// `true` if the slice is contiguous within the `CMBlockBuffer`,
`false`
    /// otherwise. Also returns `false` if the `CMBlockBuffer` is empty.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var isContiguous  Bool   get


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
public struct CMSync   Sendable


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMSync

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct Error   Sendable

        public static let missingRequiredParameter  NSError

        public static let invalidParameter  NSError

        public static let allocationFailed  NSError

        public static let rateMustBeNonZero  NSError



@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
public protocol CMSyncProtocol   Sendable

    /// Queries the relative rate of one timebase or clock relative to another
    /// timebase or clock.
    ///
    /// If both have a common master, this calculation is performed purely based
    /// on the rates in the common tree rooted in that master.
    ///
    /// If they have different master clocks (or are both clocks), this
    /// calculation takes into account the measured drift between the two clocks,
    /// using host time as a pivot.
    ///
    /// The rate of a moving timebase relative to a stopped timebase is a NaN.
    ///
    /// Calling `timebase.effectiveRate` is equivalent to calling
    /// `timebase.rate(relativeTo
timebase.ultimateMasterClock)`
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    func rate T                             T     Double
where T   CMSyncProtocol

    /// Queries the relative rate of one timebase or clock relative to another
    /// timebase or clock and the times of each timebase or clock at which the
    /// relative rate went into effect.
    ///
    /// If both have a common master, this calculation is performed purely based
    /// on the rates in the common tree rooted in that master.
    ///
    /// If they have different master clocks (or are both clocks), this
    /// calculation takes into account the measured drift between the two clocks,
    /// using host time as a pivot.
    ///
    /// The rate of a moving timebase relative to a stopped timebase is a NaN.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    func rateAndAnchorTime T                                  T
throws           Double                CMTime
CMTime  where T   CMSyncProtocol

    /// Converts a time from one timebase or clock to another timebase or clock.
    ///
    /// If both have a common master, this calculation is performed purely based
    /// on the mathematical rates and offsets in the common tree rooted in that
    /// master.
    ///
    /// If they have different master clocks (or are both clocks), this
    /// calculation also compensates for measured drift between the clocks.
    ///
    /// To convert to or from host time, use `CMClock.hostTimeClock`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    func convertTime T  _       CMTime                      T
   CMTime where T   CMSyncProtocol

    /// Reports whether it is possible for one timebase or clock to drift relative
    ///  to the other.
    ///
    ///  A timebase can drift relative to another if they are ultimately mastered
    ///  by clocks that can drift relative to each other.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    func mightDrift T                              T     Bool
where T   CMSyncProtocol

    ///  Time from a clock or timebase.
    ///
    ///  `time` simply calls either `CMClock.time` or `CMTimebase.time`,
as
    ///  appropriate.
    ///
    ///  It comes in handy for things like:
    ///  ```
    ///  let master = timebase.master
    ///  let time = master.time
    ///  ```
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    var time  CMTime   get


@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
public func CMTIMERANGE_IS_EMPTY _        CMTimeRange     Bool

@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
public func CMTIMERANGE_IS_INDEFINITE _        CMTimeRange
Bool

@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
public func CMTIMERANGE_IS_INVALID _        CMTimeRange
Bool

@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
public func CMTIMERANGE_IS_VALID _        CMTimeRange     Bool

@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
public func CMTIME_HAS_BEEN_ROUNDED _        CMTime     Bool

@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
```

```
public func CMTIME_IS_INDEFINITE _       CMTime     Bool

@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
public func CMTIME_IS_INVALID _       CMTime     Bool

@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
public func CMTIME_IS_NEGATIVEINFINITY _       CMTime     Bool

@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
public func CMTIME_IS_NUMERIC _       CMTime     Bool

@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
public func CMTIME_IS_POSITIVEINFINITY _       CMTime     Bool

@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
public func CMTIME_IS_VALID _       CMTime     Bool

/**
      CMTag is used to label something about a resource or other media construct.
A CMTag contains a category and a value to represent a particular tag that might be
assigned to or associated with another resource.  There is only one of each of the
category and the value so any notion of "has" is about the respective singular
element.  CMTags carry a single value that can be carried in 64 bits.  This can
include data types such as signed 64-bit integers, floating point values fitting in 64
bits, up to 64 bit of flags, and other data types fitting within 64 bits.  A CMTag value
should not be used to carry pointers or objects.  If such a reference is needed, it is
okay to carry an index into an out-of-band data structure that itself has a memory
reference or an object reference.
 */
@available macOS 14.0  iOS 17.0  tvOS 17.0  watchOS 10.0
visionOS 1.0
public class CMTag   Equatable  CustomStringConvertible
@unchecked Sendable

    public typealias RawCategory   FourCharCode

    /**
          The Value enum encapsulates the type and holds the value for the tag.
     */
    public enum Value   Sendable  Equatable

        case int64 Int64

        case float64 Float64

        case osType OSType
        case flags UInt64

        /// Returns a Boolean value indicating whether two values are equal.
        ///
        /// Equality is the inverse of inequality. For any values `a` and `b`,
        /// `a == b` implies that `a != b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to compare.
        public static func        CMTag Value     CMTag Value
    Bool


    /**
        The category for the tag.
     */
    final public let rawCategory  CMTag RawCategory

    /**
        The value for the tag.
     */
    final public let rawTagValue  CMTag Value

    /**
     Initializes a CMTag instance with the specified category and value.

     - Parameters:
        - category: The category to use for the CMTag.
        - rawTagValue: The value to use for the CMTag.
    */
    public init              CMTag RawCategory
CMTag Value

    /**
     Returns the strongly typed value for a tag if it matches the specified category.
Returns nil if the category of the tag doesn't match the specified category.

     - Parameters:
     - category: The category to match.
     */
    public func value T
CMTypedTag T  Category     T  where T   Sendable

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
    public static func          CMTag        CMTag      Bool

    /// A textual representation of this instance.
    ///
    /// Calling this property directly is discouraged. Instead, convert an
    /// instance of any type to a string by using the `String(describing:)`
    /// initializer. This initializer works with any type, and uses the custom
    /// `description` property for types that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point: CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string in the assignment to `s` uses the
    /// `Point` type's `description` property.
    public var description  String   get


@available macOS 14.0  iOS 17.0  tvOS 17.0  watchOS 10.0
visionOS 1.0
extension CMTag

    /**
     Initializes a CMTag instance for a mediaType tag with the specified value.

     - Parameters:
     - value:  The value for the tag.
     */
    public static func mediaType _
CMFormatDescription MediaType
CMTypedTag CMFormatDescription MediaType

    /**
     Initializes a CMTag instance for a mediaSubType tag with the specified value.

     - Parameters:
     - value:  The value for the tag.
```

```
      */
    public static func mediaSubType _
CMFormatDescription MediaSubType
CMTypedTag CMFormatDescription MediaSubType

    /**
      Initializes a CMTag instance for a trackID tag with the specified value.

      - Parameters:
      - value: The value for the tag.
      */
    public static func trackID _        CMPersistentTrackID
   CMTypedTag CMPersistentTrackID

    /**
      Initializes a CMTag instance for a channelID tag with the specified value.

      - Parameters:
      - value: The value for the tag.
      */
    public static func channelID _        Int64
CMTypedTag Int64

    /**
      Initializes a CMTag instance for a videoLayerID tag with the specified value.

      - Parameters:
      - value: The value for the tag.
      */
    public static func videoLayerID _        Int64
CMTypedTag Int64

    /**
      Initializes a CMTag instance for a pixelFormat tag with the specified value.

      - Parameters:
      - value: The value for the tag.
      */
    public static func pixelFormat _        OSType
CMTypedTag OSType

    /**
      Initializes a CMTag instance for a packingType tag with the specified value.

      - Parameters:
      - value: The value for the tag.
      */
    public static func packingType _        CMPackingType
CMTypedTag CMPackingType
```

```
    /**
      Initializes a CMTag instance for a projectionType tag with the specified value.

      - Parameters:
      - value:  The value for the tag.
      */
    public static func projectionType _
CMProjectionType     CMTypedTag CMProjectionType

    /**
      Initializes a CMTag instance for a stereoView tag with the specified value.

      - Parameters:
      - value:  The value for the tag.
      */
    public static func stereoView _
CMStereoViewComponents     CMTypedTag CMStereoViewComponents

    /**
      Initializes a CMTag instance for a stereoViewInterpretation tag with the
specified value.

      - Parameters:
      - value:  The value for the tag.
      */
    public static func stereoViewInterpretation _
CMStereoViewInterpretationOptions
CMTypedTag CMStereoViewInterpretationOptions


/**
  CMTaggedBuffer contains an array of CMTags and a buffer.
  */
@available macOS 14.0  iOS 17.0  tvOS 17.0  watchOS 10.0
visionOS 1.0
public struct CMTaggedBuffer   CustomStringConvertible

    /**
      Buffer contains an the buffer associated with the array of tags.
      */
    public enum Buffer

        /**
             A CMSampleBuffer.
             */
        case sampleBuffer CMSampleBuffer

        /**
             A CVPixelBuffer.
             */
        case pixelBuffer CVPixelBuffer


    /**
     A tag array associated with the buffer.
     */
    public let tags   CMTag

    /**
     Buffer associated with the tags.
     */
    public let buffer  CMTaggedBuffer Buffer

    /**
     Initializes a CMTaggedBuffer instance with the specified tags and buffer.

     - Parameters:
        - tags:  The tags to use.
        - buffer: The buffer to use.
    */
    public init        CMTag           CMTaggedBuffer Buffer

    /**
     Initializes a CMTaggedBuffer instance with the specified tags and a
CMSampleBuffer.

     - Parameters:
        - tags:  The tags to use.
        - sampleBuffer: The sample buffer to use.
    */
    public init        CMTag                 CMSampleBuffer

    /**
     Initializes a CMTaggedBuffer instance with the specified tags and a
CVPixelBuffer.

     - Parameters:
        - tags:  The tags to use.
        - pixelBuffer: The pixel buffer to use.
    */
    public init        CMTag                CVPixelBuffer

    /// A textual representation of this instance.
    ///
    /// Calling this property directly is discouraged. Instead, convert an
    /// instance of any type to a string by using the `String(describing:)`
    /// initializer. This initializer works with any type, and uses the custom
    /// `description` property for types that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point: CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string in the assignment to `s` uses the
    /// `Point` type's `description` property.
    public var description  String    get


/// Deprecated synonym
@available                    10.8                 12.0
"CMTimebaseSetAnchorTime(_:timebaseTime:immediateSourceTime:)"

@available                  6.0                 15.0
"CMTimebaseSetAnchorTime(_:timebaseTime:immediateSourceTime:)"

@available                   9.0                 15.0
"CMTimebaseSetAnchorTime(_:timebaseTime:immediateSourceTime:)"

@available                      6.0                 8.0
"CMTimebaseSetAnchorTime(_:timebaseTime:immediateSourceTime:)"

public func CMTimebaseSetAnchorTime _           CMTimebase
              CMTime                          CMTime

/// Deprecated synonym
@available                    10.8                 12.0
"CMTimebaseSetRateAndAnchorTime(_:rate:anchorTime:immediateSou
rceTime:)"
@available                  6.0                 15.0
"CMTimebaseSetRateAndAnchorTime(_:rate:anchorTime:immediateSou
rceTime:)"
@available                   9.0                 15.0
"CMTimebaseSetRateAndAnchorTime(_:rate:anchorTime:immediateSou
rceTime:)"
@available                      6.0                 8.0
"CMTimebaseSetRateAndAnchorTime(_:rate:anchorTime:immediateSou
rceTime:)"
public func CMTimebaseSetRateAndAnchorTime _
CMTimebase         Double               CMTime
```

```
                        CMTime

/**
        CMTypedTag contains strongly typed categories and values.  CMTypedTag
enforces the defined data types for specific categories.

        A custom tag type can be created by following the CustomColor pattern below.
        1.  Create an extension for CMTypedTag.Category specialized to the new type.
Implement the init function and create the valueForTagValue and tagValueForValue
closures to map the values.
        2.  Create an extension on CMTypedTag.Category to create the strongly typed
category.
        3.  Create an extension on CMTag to create a new tag with the new category
and strongly typed value.

        internal extension CMTypedTag.Category where TypedValue == CustomColor {
                init(rawCategory: RawCategory) {
                        self.init(rawCategory: rawCategory, valueForTagValue: { tagValue in
                                if case let .int64(value) = tagValue {
                                        return CustomColor.init(rawValue: value)
                                }

                                return nil
                        }, tagValueForValue: { value in
                                return .int64(value.rawValue)
                        })
                }
        }

        extension CMTypedTag.Category {
                public static var customColor: CMTypedTag<CustomColor>.Category
{ .init(rawCategory: .init(string: "colr")) }
        }

        extension CMTag {
                public static func customColor(_ value: CustomColor) ->
CMTypedTag<CustomColor> { .init(category: .customColor, value: value) }
        }
 */
@available macOS 14.0  iOS 17.0  tvOS 17.0  watchOS 10.0
visionOS 1.0
public class CMTypedTag TypedValue    CMTag where TypedValue
Sendable


    /**
      Category is strongly typed to the same expected type of the value.  The
Category contains functions for converting between the typed value and the CMTag's
value.
      */
    public struct Category    Sendable

        /**
                The low-level category.
                */
        public let rawCategory
CMTypedTag TypedValue  RawCategory

        /**
             A function to get the value for a specific CMTag's value.
             */
        public func value               CMTag Value
TypedValue

        /**
             A function to get the CMTag's value for a specific typed value.
             */
        public func tagValue            TypedValue
CMTag Value

        /**
             Initializes a Category instance with the specified low-level category
and closures to convert to/from the CMTag's value and the typed tag value.

             - Parameters:
             - rawCategory:  The category to use for the CMTag.
             - valueForTagValue: A closure to convert from the CMTag's value
to the typed value.
             - tagValueForValue: A closure to convert from the typed value to
the CMTag's value.
             */
        public init
CMTypedTag TypedValue  RawCategory
@escaping @Sendable  CMTag Value     TypedValue
                  @escaping @Sendable  TypedValue
CMTag Value


    /**
     The strongly typed category for the tag.
     */
    final public let category  CMTypedTag TypedValue  Category

    /**
     The strongly typed value for the tag.
     */
    public var value  TypedValue   get

    /**
     Initializes a CMTypedTag instance with the specified category and value.

     - Parameters:
        - category:  The category to use for the CMTag.
        - value: The value to use for the tag.
    */
    public init           CMTypedTag TypedValue  Category
```

```
        TypedValue


/**
  The predefined categories with strongly typed values.
  */
@available macOS 14.0  iOS 17.0  tvOS 17.0  watchOS 10.0
visionOS 1.0
extension CMTypedTag Category

    /**
      @ A category representing a media type.  The value is a
CMFormatDescription.MediaType.
      */
    public static var mediaType
CMTypedTag CMFormatDescription MediaType  Category   get

    /**
      @ A category representing a media sub type.  The value is a
CMFormatDescription.MediaSubType
      */
    public static var mediaSubType
CMTypedTag CMFormatDescription MediaSubType  Category   get

    /**
      @ A category representing a track id.  The value is a CMPersistentTrackID for
a corresponding asset.
      */
    public static var trackID
CMTypedTag CMPersistentTrackID  Category   get

    /**
      @ A category representing a channel id.  The value is the
CMVideoTarget/CMVideoReceiver channel identifier.
      */
    public static var channelID  CMTypedTag Int64  Category
get

    /**
      @ A category representing a video layer id.  The value is a signed 64-bit
integer specifying the video layer identifier.
      */
    public static var videoLayerID  CMTypedTag Int64  Category
  get

    /**
      @ A category representing a pixel format.  The value is the pixel format of the
buffer or channel, if pixel-based, corresponding to a pixel format (i.e., a
kCVPixelFormatType_* type).
      */
    public static var pixelFormat  CMTypedTag OSType  Category
  get

    /**
      @ A category representing a packing type.  The value is a CMPackingType
indicating this channel is packed in some way (e.g., frame packed, texture atlas).
      */
    public static var packingType
CMTypedTag CMPackingType  Category   get

    /**
      @ A category representing a projection] type.  The value is a
CMProjectionType indicating textures are related to a kind of texture projection (e.g.,
equirectangular).
      */
    public static var projectionType
CMTypedTag CMProjectionType  Category   get

    /**
      @ A category representing a stereo view type.  The value is a
CMStereoViewComponents indicating this channel is related to carrying
stereographic views.
      */
    public static var stereoView
CMTypedTag CMStereoViewComponents  Category   get

    /**
      @ A category representing a stereo view interpretation type.  The value is a
CMStereoViewInterpretationOptions indicating this channel has non default stereo
view interpretation (e.g., stereo eye view order is reversed.)  Tags with this category
will typically be associated with tags of category stereoView.  This tag alone however
does not indicate which stereo eyes are present.
      */
    public static var stereoViewInterpretation
CMTypedTag CMStereoViewInterpretationOptions  Category   get


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMClock

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct Error   Sendable

        public static let missingRequiredParameter  NSError

        public static let invalidParameter  NSError

        public static let allocationFailed  NSError

        public static let unsupportedOperation  NSError
```

```
@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMTimebase

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct Error   Sendable

        public static let missingRequiredParameter  NSError

        public static let invalidParameter  NSError

        public static let allocationFailed  NSError

        public static let timerIntervalTooShort  NSError

        public static let readOnly  NSError



@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMClock

    /// The `CFTypeID` corresponding to `CMClock`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static var typeID  CFTypeID   get

    /// The singleton clock logically identified with host time.
    ///
    /// On Mac OS X, the host time clock uses `mach_absolute_time` but
returns a
    /// value with a large integer timescale (eg, nanoseconds).
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static var hostTimeClock  CMClock   get

    /// Converts a host time from `CMTime` to the host time's native units.
    ///
    /// This function performs a scale conversion, not a clock conversion.
    ///
    /// It can be more accurate than `CMTimeConvertScale` because the
system units
    /// may have a non-integer timescale.
    ///
    /// On Mac OS X, this function converts to the units of
```

```
`mach_absolute_time`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static func convertHostTimeToSystemUnits _
          CMTime     UInt64

    /// Converts a host time from native units to `CMTime`.
    ///
    /// The returned value will have a large integer timescale (eg, nanoseconds).
    ///
    /// This function handles situations where host time's native units use a
    /// non-integer timescale.
    ///
    /// On Mac OS X, this function converts from the units of
`mach_absolute_time`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static func convertSystemUnitsToHostTime _
             UInt64     CMTime

    /// The current time.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var time  CMTime   get

    /// Retrieves the current time from a clock and also the matching time from
    /// the clock's reference clock.
    ///
    ///  To make practical use of this, you may need to know what the clock's
    /// reference clock is.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func anchorTime   throws                 CMTime
               CMTime

    /// Indicates whether it is possible for the clock to drift relative to the
    /// `otherClock`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func mightDrift                        CMClock
Bool

    /// Makes the clock stop functioning.
    ///
    /// After invalidation, the clock will return errors from all APIs.
    ///
    /// This should only be called by the "owner" of the clock, who knows (for
    /// example) that some piece of hardware has gone away, and the clock will
no
```

```
    /// longer work (and might even crash).
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func invalidate


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMTimebase

    /// The `CFTypeID` corresponding to `CMTimebase`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static var typeID  CFTypeID   get

    /// The immediate source timebase.
    ///
    /// Returns `nil` if the timebase actually has a source clock instead of a
    /// source timebase.
    @available macOS 12.0  iOS 15.0  tvOS 15.0  watchOS 8.0
visionOS 1.0
    public var sourceTimebase  CMTimebase    get

    /// Deprecated synonym for sourceTimebase.
    @available                    10.15                12.0
         "sourceTimebase"
    @available                   13.0                15.0
         "sourceTimebase"
    @available                    13.0                15.0
         "sourceTimebase"
    @available                      6.0                8.0
         "sourceTimebase"
    public var masterTimebase  CMTimebase    get

    /// Returns the immediate source clock.
    ///
    /// Returns `nil` if the timebase actually has a source timebase instead of
a
    /// source clock.
    @available macOS 12.0  iOS 15.0  tvOS 15.0  watchOS 8.0
visionOS 1.0
    public var sourceClock  CMClock    get

    /// Deprecated synonym for sourceClock.
    @available                    10.15                12.0
         "sourceClock"
    @available                   13.0                15.0
         "sourceClock"
    @available                    13.0                15.0
```

```
        "sourceClock"
    @available                      6.0               8.0
        "sourceClock"
    public var masterClock  CMClock    get

    /// Returns the immediate source (either timebase or clock).
    @available macOS 12.0  iOS 15.0  tvOS 15.0  watchOS 8.0
visionOS 1.0
    public var source  any CMSyncProtocol

    /// Deprecated synonym for source.
    @available                    10.15             12.0
        "source"
    @available                  13.0              15.0
        "source"
    @available                   13.0              15.0
        "source"
    @available                      6.0               8.0
        "source"
    public var master  any CMSyncProtocol

    /// The source clock that is the source of all of the source timebases.
    @available macOS 12.0  iOS 15.0  tvOS 15.0  watchOS 8.0
visionOS 1.0
    public var ultimateSourceClock  CMClock   get

    /// Deprecated synonym for ultimateSourceClock.
    @available                    10.15             12.0
        "ultimateSourceClock"
    @available                  13.0              15.0
        "ultimateSourceClock"
    @available                   13.0              15.0
        "ultimateSourceClock"
    @available                      6.0               8.0
        "ultimateSourceClock"
    public var ultimateMasterClock  CMClock   get

    /// The current time.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var time  CMTime   get

    /// Retrieves the current time in the specified timescale.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func time                              CMTimeScale
         CMTimeRoundingMethod                       CMTime

    /// Sets the current time.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func setTime _        CMTime  throws

    /// Sets the time at a particular master time.
    ///
    /// `time` will be interpolated from that anchor time.
    ///
    /// `timebase.setTime(time)` is equivalent to calling
    /// `timebase.setAnchorTime(time, timebase.master.time)`
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func setAnchorTime _              CMTime
                CMTime  throws

    /// The current rate.
    ///
    /// This is the rate relative to its immediate master clock or timebase.
    /// For example, if a timebase is running at twice the rate of its master,
    /// its rate is 2.0.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var rate  Double   get

    /// The current time and rate.
    ///
    /// You can use this to take a consistent snapshot of the two values,
    /// avoiding possible inconsistencies due to external changes between
    /// retrieval of `time` and `rate`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var timeAndRate         CMTime        Double    get


    /// Sets the rate.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func setRate _        Double  throws

    /// Sets the time at a particular master time, and changes the rate at exactly
    /// that time.
    ///
    /// `time` will be interpolated from that anchor time as though the timebase
    /// has been running at the requested rate since that time.
    /// `timebase.setRate(rate)` is approximately equivalent to calling
    /// ```
    /// timebase.setRateAndAnchorTime(rate: rate,
    ///                               anchorTime:
timebase.time,
```

```
    ///                              referenceTime:
timebase.master.time)
    /// ```
    /// except that `setRate` will not generate a `TimeJumped` notification,
and
    /// `setRateAndAnchorTime` will.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func setRateAndAnchorTime       Double
CMTime                 CMTime  throws

    /// The effective rate (which combines its rate with the rates of all its
    /// master timebases).
    ///
    /// Calling `timebase.effectiveRate` is equivalent to calling
    /// `timebase.rate(relativeTo:
timebase.ultimateMasterClock)`
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var effectiveRate  Double   get

    /// Adds the timer to the list of timers managed by the timebase.
    ///
    /// The timer must be a repeating run loop timer (with a very long interval at
    /// least as long as .veryLongTimeInterval), attached to a runloop.
    ///
    /// The timebase will retain the timer, and will maintain its "NextFireDate"
    /// according to the `CMTime` set using `setTimerNextFireTime`.
    ///
    /// Until the first call to `setTimerNextFireTime`, the "NextFireDate" will
be
    /// set far, far in the future. The runloop that timer is attached to must be
    /// passed in and the timebase will retain that runloop. The retained runloop
    /// will be used to call `CFRunLoopWakeUp()` any time the timebase
modifies
    /// the timer's fire date.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func addTimer _        Timer              RunLoop
throws

    /// Quite a while (256 years).
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static let veryLongTimeInterval  CFTimeInterval

    /// Quite a while from 2001 (2257).
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static let farFuture  CFAbsoluteTime
```

```
    /// Removes the timer from the list of timers managed by the timebase.
    ///
    /// The timebase will no longer maintain the timer's "NextFireDate".
    ///
    /// If the timer is invalidated, the timebase will eventually remove it from
    /// its list and release it even if this function is not called.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func removeTimer _         Timer  throws

    /// Sets the `CMTime` on the timebase's timeline at which the timer should
    /// next be fired.
    ///
    /// The timer must be on the list of timers managed by the timebase.
    ///
    /// The timebase will continue to update the timer's "NextFireDate" according
    /// to time jumps and effective rate changes.
    ///
    /// If `fireTime` is not numeric, or if the timebase is not moving, the
    /// "NextFireDate" will be set to a date far, far in the future.
    ///
    /// IMPORTANT NOTE: Due to the way that `CFRunLoopTimer`s are
implemented, if
    /// a timer passes through a state in which it is due to fire, it may fire
    /// even if its rescheduled before the runloop runs again. Clients should take
    /// care to avoid temporarily scheduling timers in the past. For example, set
    /// the timebase's rate or time before you set the timer's next fire time, if
    /// you are doing both at once. (If setting the timebase's rate or time might
    /// put the timer's fire time in the past, you may need to set the fire time
    /// to `CMTime.invalid` across the timebase change.)
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func setTimerNextFireTime _         Timer
CMTime  throws

    /// Sets the timer to fire immediately once, overriding any previous
    /// `setTimerNextFireTime` call.
    ///
    /// The timer must be on the list of timers managed by the timebase.
    ///
    /// This is equivalent to calling
    /// `CFRunLoopTimerSetNextFireDate(timer,
CFAbsoluteTimeGetCurrent())`
    /// except that the timebase gets to know that it shouldn't interfere.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func setTimerToFireImmediately _         Timer
throws
```

```
    /// Adds the timer dispatch source to the list of timers managed by the
    /// timebase.
    ///
    /// The timer source must have been created by calling
    /// `DispatchSource.makeTimerSource(flags: [], queue:
some_dispatch_queue)`
    /// and should have had an event handler associated with it via
    /// `timerSource.setEventHandler { /* timer fired */ }
    /// Don't forget to call `timerSource.activate()` as dispatch sources
are
    /// created in an inactive state.
    ///
    /// The timebase will retain the timer source, and will maintain its start
    /// time according to the `CMTime` set using `setTimerNextFireTime`.
    ///
    /// Until the first call to `setTimerNextFireTime`, the start time will be
set
    /// to `DispatchTime.distantFuture`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func addTimer T  _         T  throws where T
DispatchSourceTimer

    /// Removes the timer dispatch source from the list of timers managed by the
    /// timebase.
    ///
    /// The timebase will no longer maintain the timer source's start time.
    ///
    /// If the timer source is cancelled, the timebase will eventually remove it
    /// from its list and release it even if this function is not called.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func removeTimer T  _         T  throws where T
DispatchSourceTimer

    /// Sets the `CMTime` on the timebase's timeline at which the timer dispatch
    /// source should next be fired.
    ///
    /// The timer source must be on the list of timers managed by the timebase.
    ///
    /// The timebase will continue to update the timer dispatch source's start
    /// time according to time jumps and effective rate changes.
    ///
    /// If `fireTime` is not numeric, or if the timebase is not moving, the start
    /// time will be set to `DispatchTime.distantFuture`.
    ///
    /// IMPORTANT NOTE: Due to the way that timer dispatch sources are
    /// implemented, if a timer passes through a state in which it is due to fire,
    /// it may fire even if its rescheduled before the event handler is run.
    ///
    /// Clients should take care to avoid temporarily scheduling timers in the
    /// past. For example, set the timebase's rate or time before you set the
    /// timer's next fire time, if you are doing both at once. (If setting the
    /// timebase's rate or time might put the timer's fire time in the past, you
    /// may need to set the fire time to `CMTime.invalid` across the
timebase
    /// change.)
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func setTimerNextFireTime T  _         T
CMTime  throws where T   DispatchSourceTimer

    /// Sets the timer dispatch source to fire immediately once, overriding any
    /// previous `setTimerNextFireTime` call.
    ///
    /// The timer source must be on the list of timers managed by the timebase.
    ///
    /// This is equivalent to calling
    /// `timerSource.schedule(deadline: DispatchTime.now())`
    /// except that the timebase gets to know that it shouldn't interfere.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func setTimerToFireImmediately T  _         T
throws where T   DispatchSourceTimer

    /// Requests that the timebase wait until it is not posting any notifications.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func notificationBarrier   throws


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMTimebase

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static let effectiveRateChanged
NSNotification Name

    /// Posted by a timebase after a discontinuous time jump.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static let timeJumped  NSNotification Name

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct NotificationKey   @unchecked Sendable
```

```
        /// The raw type that can be used to represent all values of the
conforming
        /// type.
        ///
        /// Every distinct value of the conforming type has a corresponding
unique
        /// value of the `RawValue` type, but there may be values of the
`RawValue`
        /// type that don't have a corresponding value of the conforming type.
        public typealias RawValue   CFString

        /// The corresponding value of the raw type.
        ///
        /// A new instance initialized with `rawValue` will be equivalent to this
        /// instance. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter, Legal
        ///     }
        ///
        ///     let selectedSize = PaperSize.Letter
        ///     print(selectedSize.rawValue)
        ///     // Prints "Letter"
        ///
        ///     print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue)!)
        ///     // Prints "true"
        public var rawValue  CFString

        /// Creates a new instance with the specified raw value.
        ///
        /// If there is no value of the type that corresponds with the specified
raw
        /// value, this initializer returns `nil`. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter, Legal
        ///     }
        ///
        ///     print(PaperSize(rawValue: "Legal"))
        ///     // Prints "Optional("PaperSize.Legal")"
        ///
        ///     print(PaperSize(rawValue: "Tabloid"))
        ///     // Prints "nil"
        ///
        /// - Parameter rawValue: The raw value to use for the new
instance.
        public init           CFString
        ///  Payload key for the time at which a change in effective rate or a
        ///  discontinuous time jump occurred.
        @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS
6.0  visionOS 1.0
        public static let eventTime
CMTimebase NotificationKey



@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMTimebase   CMSyncProtocol

    ///  Queries the relative rate of one timebase or clock relative to another
    ///  timebase or clock.
    ///
    ///  If both have a common master, this calculation is performed purely based
    ///  on the rates in the common tree rooted in that master.
    ///
    ///  If they have different master clocks (or are both clocks), this
    ///  calculation takes into account the measured drift between the two clocks,
    ///  using host time as a pivot.
    ///
    ///  The rate of a moving timebase relative to a stopped timebase is a NaN.
    ///
    ///  Calling `timebase.effectiveRate` is equivalent to calling
    ///  `timebase.rate(relativeTo
timebase.ultimateMasterClock)`
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func rate T
T     Double where T   CMSyncProtocol

    ///  Queries the relative rate of one timebase or clock relative to another
    ///  timebase or clock and the times of each timebase or clock at which the
    ///  relative rate went into effect.
    ///
    ///  If both have a common master, this calculation is performed purely based
    ///  on the rates in the common tree rooted in that master.
    ///
    ///  If they have different master clocks (or are both clocks), this
    ///  calculation takes into account the measured drift between the two clocks,
    ///  using host time as a pivot.
    ///
    ///  The rate of a moving timebase relative to a stopped timebase is a NaN.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func rateAndAnchorTime T
                 T  throws           Double
```

```
CMTime                    CMTime  where T   CMSyncProtocol

    /// Converts a time from one timebase or clock to another timebase or clock.
    ///
    /// If both have a common master, this calculation is performed purely based
    /// on the mathematical rates and offsets in the common tree rooted in that
    /// master.
    ///
    /// If they have different master clocks (or are both clocks), this
    /// calculation also compensates for measured drift between the clocks.
    ///
    /// To convert to or from host time, use `CMClock.hostTimeClock`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func convertTime T  _        CMTime
                 T      CMTime where T   CMSyncProtocol

    /// Reports whether it is possible for one timebase or clock to drift relative
    /// to the other.
    ///
    /// A timebase can drift relative to another if they are ultimately mastered
    /// by clocks that can drift relative to each other.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func mightDrift T                            T
   Bool where T   CMSyncProtocol


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMClock   CMSyncProtocol

    /// Queries the relative rate of one timebase or clock relative to another
    /// timebase or clock.
    ///
    /// If both have a common master, this calculation is performed purely based
    /// on the rates in the common tree rooted in that master.
    ///
    /// If they have different master clocks (or are both clocks), this
    /// calculation takes into account the measured drift between the two clocks,
    /// using host time as a pivot.
    ///
    /// The rate of a moving timebase relative to a stopped timebase is a NaN.
    ///
    /// Calling `timebase.effectiveRate` is equivalent to calling
    /// `timebase.rate(relativeTo
timebase.ultimateMasterClock)`
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
```

```
    public func rate T
T     Double where T   CMSyncProtocol

    /// Queries the relative rate of one timebase or clock relative to another
    /// timebase or clock and the times of each timebase or clock at which the
    /// relative rate went into effect.
    ///
    /// If both have a common master, this calculation is performed purely based
    /// on the rates in the common tree rooted in that master.
    ///
    /// If they have different master clocks (or are both clocks), this
    /// calculation takes into account the measured drift between the two clocks,
    /// using host time as a pivot.
    ///
    /// The rate of a moving timebase relative to a stopped timebase is a NaN.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func rateAndAnchorTime T
                  T  throws           Double
CMTime                      CMTime  where T   CMSyncProtocol

    /// Converts a time from one timebase or clock to another timebase or clock.
    ///
    /// If both have a common master, this calculation is performed purely based
    /// on the mathematical rates and offsets in the common tree rooted in that
    /// master.
    ///
    /// If they have different master clocks (or are both clocks), this
    /// calculation also compensates for measured drift between the clocks.
    ///
    /// To convert to or from host time, use `CMClock.hostTimeClock`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func convertTime T  _        CMTime
                  T     CMTime where T   CMSyncProtocol

    /// Reports whether it is possible for one timebase or clock to drift relative
    /// to the other.
    ///
    /// A timebase can drift relative to another if they are ultimately mastered
    /// by clocks that can drift relative to each other.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func mightDrift T                              T
   Bool where T   CMSyncProtocol


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
```

```
extension CMBlockBuffer   CMAttachmentBearerProtocol

    /// Access attachments.
    public var attachments  CMAttachmentBearerAttachments

    /// Copy all propagatable attachments from one buffer to another.
    ///
    /// - Parameter destination: `CMAttachmentBearerProtocol`
object to copy
    ///     attachments to.
    public func propagateAttachments T                  T
where T   CMAttachmentBearerProtocol


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMSampleBuffer   CMAttachmentBearerProtocol

    /// Access attachments.
    public var attachments  CMAttachmentBearerAttachments

    /// Copy all propagatable attachments from one buffer to another.
    ///
    /// - Parameter destination: `CMAttachmentBearerProtocol`
object to copy
    ///     attachments to.
    public func propagateAttachments T                  T
where T   CMAttachmentBearerProtocol


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CVBuffer   CMAttachmentBearerProtocol

    /// Access attachments.
    public var attachments  CMAttachmentBearerAttachments

    /// Copy all propagatable attachments from one buffer to another.
    ///
    /// - Parameter destination: `CMAttachmentBearerProtocol`
object to copy
    ///     attachments to.
    public func propagateAttachments T                  T
where T   CMAttachmentBearerProtocol


extension CMClock

    /// Changes the CoreAudio device the clock is tracking.
    ///
    /// Pass `nil` for `deviceUID` to make the clock track the default device.
    @available macOS 10.15  macCatalyst 13.0
    @available
    @available
    @available
    @available
    public func setAudioDeviceUID _            String   throws

    /// Changes the CoreAudio device the clock is tracking.
    @available macOS 10.15  macCatalyst 13.0
    @available
    @available
    @available
    @available
    public func setAudioDeviceID _           AudioDeviceID
throws

    /// Queries which CoreAudio device the clock is tracking.
    ///
    /// If a non-`nil` `deviceUID` has been set, `audioDevice()` returns
the set
    /// UID, its associated ID, and `trackingDefaultDevice` == false.
    ///
    /// If a `deviceID` has been set directly, `audioDevice()` returns
`nil` UID,
    /// the set device ID, and `trackingDefaultDevice` == false.
    ///
    /// If a `nil` `deviceUID` has been set (which means "track the default
    /// device"), `audioDevice()` returns `nil` UID, the ID of the current
default
    /// device, and `trackingDefaultDevice` == true.
    @available macOS 10.15  macCatalyst 13.0
    @available
    @available
    @available
    @available
    public func audioDevice   throws                String
          AudioDeviceID                    Bool


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMSimpleQueue

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct Error   Sendable

        /// An allocation failed.
        public static let allocationFailed  NSError
        /// `nil` or `0` was passed for a required parameter.
        public static let requiredParameterMissing  NSError

        /// An out-of-range value was passed for a parameter with a restricted
valid
        /// range.
        public static let parameterOutOfRange  NSError

        /// Operation failed because queue was full.
        public static let queueIsFull  NSError


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMSimpleQueue

    /// The `CFTypeID` corresponding to `CMSimpleQueue`.
    ///
    /// You can check if a CFTypeRef object is actually a CMSimpleQueue by
    /// comparing CFGetTypeID(object) with CMSimpleQueue.typeID.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static var typeID  CFTypeID   get

    /// Enqueues an element on the queue.
    ///
    /// If the queue is full, this operation will fail.
    ///
    /// - Parameter element:  Element to enqueue.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func enqueue _          UnsafeRawPointer  throws

    /// Dequeues an element from the queue.
    ///
    /// - Returns:  The dequeued element. nil if the queue was empty, or if
there
    /// was some other error.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func dequeue      UnsafeRawPointer

    /// Returns the element at the head of the queue.
    ///
    /// - Returns:  The head element. nil if the queue was empty, or if there
was
    /// some other error.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
```

```
visionOS 1.0
    public var head  UnsafeRawPointer    get

    ///  Resets the queue.
    ///
    ///  This function resets the queue to its empty state; all values in the queue
    ///  prior to reset are lost. Note that CMSimpleQueueReset is not
synchronized
    ///  in any way, so the reader thread and writer thread must be held off by the
    ///  client during this operation.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func reset   throws

    ///  The number of elements that can be held in the queue.  Returns 0 if there
    ///  is an error.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var capacity  Int   get

    ///  The number of elements currently in the queue. Returns 0 if there is an
    ///  error.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var count  Int   get

    ///  A convenience macro that returns GetCount/GetCapacity, computed in
    ///  floating point.
    ///  0.0 is empty, 0.5 is half full, 1.0 is full.
    ///  Returns 0.0 if there is an error
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var fullness  Float   get


@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
extension CMTimeRange

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public init        CMTime        CMTime

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public var isValid  Bool   get

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
```

```
    public var isIndefinite  Bool    get

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public var isEmpty  Bool    get

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public var end  CMTime    get

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public func union _            CMTimeRange
CMTimeRange

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public func intersection _            CMTimeRange
CMTimeRange

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public func containsTime _       CMTime     Bool

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public func containsTimeRange _        CMTimeRange
Bool


@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
extension CMTimeRange   Equatable

    ///  Returns a Boolean value indicating whether two values are equal.
    ///
    ///  Equality is the inverse of inequality. For any values `a` and `b`,
    ///  `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///    - lhs:  A value to compare.
    ///    - rhs:  Another value to compare.
    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public static func             CMTimeRange
CMTimeRange     Bool

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
```

```
    public static func              CMTimeRange
CMTimeRange     Bool


@available macOS 13.0  iOS 16.0  tvOS 16.0  watchOS 9.0
visionOS 1.0
extension CMTimeRange   Hashable

    /// Hashes the essential components of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform to the `Hashable` protocol. The
    /// components used for hashing must be the same as the components
compared
    /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your implementation of `hash(into:)`,
    ///   don't call `finalize()` on the `hasher` instance provided,
    ///   or replace it with a different instance.
    ///   Doing so may become a compile-time error in the future.
    ///
    /// - Parameter hasher: The hasher to use when combining the
components
    ///   of this instance.
    @available macOS 13.0  iOS 16.0  tvOS 16.0  watchOS 9.0
visionOS 1.0
    public func hash              inout Hasher

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable`
requirement. To
    ///   conform to `Hashable`, implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an implementation for `hashValue` for you.
    public var hashValue  Int   get


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMSampleBuffer

    /// CMFormatDescription Errors
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
```

```
    public struct Error   Sendable

        /// An allocation failed.
        public static let allocationFailed  NSError

        /// `nil` or `0` was passed for a required parameter.
        public static let requiredParameterMissing  NSError

        /// Attempt was made to set a `dataBuffer` on a
`CMSampleBuffer` that
        /// already has one.
        public static let alreadyHasDataBuffer  NSError

        /// Buffer could not be made ready.
        public static let bufferNotReady  NSError

        /// Sample index was not between `0` and `numSamples - 1`,
inclusive.
        public static let sampleIndexOutOfRange  NSError

        /// Attempt to get sample size information when there was none.
        public static let bufferHasNoSampleSizes  NSError

        /// Attempt to get sample timing information when there was none.
        public static let bufferHasNoSampleTimingInfo  NSError

        /// Output array was not large enough for the array being requested.
        public static let arrayTooSmall  NSError

        /// Timing info or size array entry count was not `0`, `1`, or
`numSamples`.
        public static let invalidEntryCount  NSError

        /// Sample buffer does not contain sample sizes.
        /// This can happen when the samples in the buffer are non-contiguous
(eg.
        /// non-interleaved audio, where the channel values for a single sample
are
        /// scattered through the buffer).
        public static let cannotSubdivide  NSError

        /// Buffer unexpectedly contains a non-numeric sample timing info.
        public static let sampleTimingInfoInvalid  NSError

        /// The media type specified by a format description is not valid for the
        /// given operation (eg. a `CMSampleBuffer` with a non-audio format
        /// description passed to
`withUnsafeAudioStreamPacketDescriptions()`).
        public static let invalidMediaTypeForOperation
NSError
```

```
        /// Buffer contains bad data. Only returned by CMSampleBuffer`
functions
        /// that inspect its sample data.
        public static let invalidSampleData  NSError

        /// The format of the given media does not match the given format
        /// description (eg. a format description paired with a
`CVImageBuffer` that
        /// fails `matchesImageBuffer()`).
        public static let invalidMediaFormat  NSError

        /// The sample buffer was invalidated.
        public static let invalidated  NSError

        /// The sample buffer's data loading operation failed (generic error).
        public static let dataFailed  NSError

        /// The sample buffer's data loading operation was canceled.
        public static let dataCanceled  NSError



@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMSampleBuffer

    /// Flags passed to various `CMSampleBuffer` APIs.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct Flags   OptionSet  Sendable

        /// The corresponding value of the raw type.
        ///
        /// A new instance initialized with `rawValue` will be equivalent to this
        /// instance. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter, Legal
        ///     }
        ///
        ///     let selectedSize = PaperSize.Letter
        ///     print(selectedSize.rawValue)
        ///     // Prints "Letter"
        ///
        ///     print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue)!)
        ///     // Prints "true"
        public let rawValue  UInt32
        /// Creates a new option set from the given raw value.
        ///
        /// This initializer always succeeds, even if the value passed as
`rawValue`
        /// exceeds the static properties declared as part of the option set. This
        /// example creates an instance of `ShippingOptions` with a raw
value beyond
        /// the highest element, with a bit mask that effectively contains all the
        /// declared static members.
        ///
        ///     let extraOptions = ShippingOptions(rawValue:
255)
        ///     print(extraOptions.isStrictSuperset(of: .all))
        ///     // Prints "true"
        ///
        /// - Parameter rawValue: The raw value of the option set to
create. Each bit
        ///   of `rawValue` potentially represents an element of the option
set,
        ///   though raw values may include bits that are not defined as distinct
        ///   values of the `OptionSet` type.
        public init           UInt32

        /// Make sure memory involved in audio buffer lists is 16-byte aligned.
        public static let
audioBufferListAssure16ByteAlignment  CMSampleBuffer Flags

        /// The type of the elements of an array literal.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias ArrayLiteralElement
CMSampleBuffer Flags

        /// The element type of the option set.
        ///
        /// To inherit all the default implementations from the `OptionSet`
protocol,
        /// the `Element` type must be `Self`, the default.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias Element   CMSampleBuffer Flags

        /// The raw type that can be used to represent all values of the
conforming
        /// type.
        ///
        /// Every distinct value of the conforming type has a corresponding
unique
        /// value of the `RawValue` type, but there may be values of the
```

```
`RawValue`
        ///  type that don't have a corresponding value of the conforming type.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias RawValue   UInt32



@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMSampleBuffer

    ///  The `CFTypeID` corresponding to `CMSampleBuffer`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static var typeID  CFTypeID   get

    ///  Associates the `CMSampleBuffer` with its `CMBlockBuffer` of
media data.
    ///
    ///  This is a write-once operation; it will fail if the `CMSampleBuffer`
    ///  already has a `dataBuffer`. This API allows a `CMSampleBuffer` to
exist,
    ///  with timing and format information, before the associated data shows up.
    ///  Example of usage: Some media services may have access to sample
size,
    ///  timing, and format information before the data is read.  Such services may
    ///  create `CMSampleBuffers` with that information and insert them into
queues
    ///  early, and use this API to attach the `CMBlockBuffer`s later, when the
    ///  data becomes ready.
    ///
    /// - Parameter dataBuffer: `CMBlockBuffer` of data being
associated.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func setDataBuffer _              CMBlockBuffer
throws

    ///  `CMBlockBuffer` of media data.
    ///
    ///  The property will be `nil` if the `CMSampleBuffer` does not contain a
    ///  `CMBlockBuffer`, if the `CMSampleBuffer` contains a
`CVImageBuffer`, or if
    ///  there is some other error.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var dataBuffer  CMBlockBuffer    get

    ///  `CVImageBuffer` of media data.
    ///
    /// The property will be `nil` if the `CMSampleBuffer` does not contain a
    /// `CVImageBuffer`, if the `CMSampleBuffer` contains a
`CMBlockBuffer`, or if
    /// there is some other error.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var imageBuffer  CVImageBuffer    get

    /// An array of `CMTaggedBuffer` of media data.
    ///
    /// The property will be `nil` if the `CMSampleBuffer` does not contain
    /// an array of CMTaggedBuffers, or if the sample buffer has been
invalidated.
    @available macOS 14.0  iOS 17.0  tvOS 17.0  watchOS 10.0
visionOS 1.0
    public var taggedBuffers   CMTaggedBuffer     get

    /// Creates a `CMBlockBuffer` containing a copy of the data from the
    /// `AudioBufferList`, and sets that as the `CMSampleBuffer`'s data
buffer.
    ///
    /// The resulting buffer(s) in the sample buffer will be 16-byte-aligned if
    /// `.audioBufferListAssure16ByteAlignment` is passed in.
    ///
    /// - Parameters:
    ///   - bufferList: Buffer list whose data will be copied into the new
    ///   `CMBlockBuffer`.
    ///   - blockBufferMemoryAllocator: Allocator to use for memory
block held by
    ///   the `CMBlockBuffer`.
    ///   - flags: Flags controlling operation.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func setDataBuffer
UnsafePointer AudioBufferList
CFAllocator
CMSampleBuffer Flags       throws

    /// Calls a closure with an `AudioBufferList` containing the data from
the
    /// `CMSampleBuffer`, and a `CMBlockBuffer` which references (and
manages the
    /// lifetime of) the data in that `AudioBufferList`. The data may or may
not
    /// be copied, depending on the contiguity and 16-byte alignment of the
    /// `CMSampleBuffer`'s data. The buffers placed in the
`AudioBufferList` are
    /// guaranteed to be contiguous. The buffers in the `AudioBufferList`
will be
```

```
    /// 16-byte-aligned if `.audioBufferListAssure16ByteAlignment` is
passed in.
    /// The `AudioBufferList` is valid only for the duration of the closure's
execution.
    ///
    /// - Parameters:
    ///   - blockBufferMemoryAllocator: Allocator to use for memory
block held by
    ///   the `CMBlockBuffer`.
    ///   - flags: Flags controlling operation.
    ///   - body: Closure to be called with a pointer to the
`AudioBufferList` and
    ///   a `CMBlockBuffer` backing the `AudioBuffer`s.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func
withAudioBufferList R
CFAllocator
CMSampleBuffer Flags
 UnsafeMutableAudioBufferListPointer  CMBlockBuffer  throws
R  throws    R

    /// Creates an array of `AudioStreamPacketDescription`s for the
variable bytes
    /// per packet or variable frames per packet audio data in the
`CMSampleBuffer`.
    ///
    /// Constant bitrate, constant frames-per-packet audio yields an empty array.
    ///
    /// This API is specific to audio format sample buffers, and will throw
    /// `.invalidMediaTypeForOperation` if called with a non-audio
sample buffer.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func audioStreamPacketDescriptions   throws
 AudioStreamPacketDescription

    /// Calls a closure with a pointer to the AudioStreamPacketDescriptions.
    ///
    /// See
`CMSampleBufferGetAudioStreamPacketDescriptionsPtr`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func withUnsafeAudioStreamPacketDescriptions R  _
       UnsafeBufferPointer AudioStreamPacketDescription
throws    R  throws    R

    /// Copies PCM audio data from the `CMSampleBuffer` into a pre-
populated
    /// `AudioBufferList`. The `AudioBufferList` must contain the
```

```
same number of
    /// channels and its data buffers must be sized to hold the specified number
    /// of frames. This API is specific to audio format sample buffers, and will
    /// throw `.invalidMediaTypeForOperation` if called with a non-audio
sample
    /// buffer. It will throw an error if the `CMSampleBuffer` does not contain
    /// PCM audio data or if its `dataBuffer` is not ready.
    ///
    /// - Parameters:
    ///   - range: Range of frames to copy.
    ///   - bufferList: Pre-populated `AudioBufferList`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func copyPCMData                         Range Int
            UnsafeMutablePointer AudioBufferList    throws

    /// A `CMSampleBuffer` data can be not-ready, ready of failed with a
status.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public enum DataReadiness    Hashable  Sendable

        /// The `CMSampleBuffer` has been created with data not ready.
        case notReady

        /// The `CMSampleBuffer` data is ready.
        case ready

        /// The `CMSampleBuffer` data will never be ready.
        case failed OSStatus

        /// Hashes the essential components of this value by feeding them into
the
        /// given hasher.
        ///
        /// Implement this method to conform to the `Hashable` protocol. The
        /// components used for hashing must be the same as the components
compared
        /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
        /// with each of these components.
        ///
        /// - Important: In your implementation of `hash(into:)`,
        ///   don't call `finalize()` on the `hasher` instance provided,
        ///   or replace it with a different instance.
        ///   Doing so may become a compile-time error in the future.
        ///
        /// - Parameter hasher: The hasher to use when combining the
components
        ///   of this instance.
        public func hash              inout Hasher

        /// Returns a Boolean value indicating whether two values are equal.
        ///
        /// Equality is the inverse of inequality. For any values `a` and `b`,
        /// `a == b` implies that `a != b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to compare.
        public static func
CMSampleBuffer DataReadiness     CMSampleBuffer DataReadiness
   Bool

        /// The hash value.
        ///
        /// Hash values are not guaranteed to be equal across different
executions of
        /// your program. Do not save hash values to use during a future
execution.
        ///
        /// - Important: `hashValue` is deprecated as a `Hashable`
requirement. To
        ///   conform to `Hashable`, implement the `hash(into:)`
requirement instead.
        ///   The compiler provides an implementation for `hashValue` for
you.
        public var hashValue  Int   get


    /// Whether or not the `CMSampleBuffer`'s data is ready or has failed.
    ///
    /// `.ready` is returned for special marker buffers, even though they have
no
    /// data.
    ///
    /// `.failed(status)` is returned if there is an error.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var dataReadiness  CMSampleBuffer DataReadiness
get

    /// Marks the `CMSampleBuffer`'s data as `.ready` or `.failed`.
    ///
    /// There is no way to undo this operation. The only way to get an "unready"
    /// `CMSampleBuffer` is to call an initializer with the `dataReady`
parameter
    /// set to `false`. Example of usage: in a read completion routine.
    ///
    /// Parameter newValue: `.ready` if the sample buffer is now ready,
    /// `.failed(status)` if the data will not become ready.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func setDataReadiness _
CMSampleBuffer DataReadiness  throws

    /// Makes the `CMSampleBuffer`'s data ready, by calling the client's
    /// `makeDataReadyHandler`.
    ///
    /// See `CMSampleBufferMakeDataReady`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func makeDataReady   throws

    /// Associates the `CMSampleBuffer`'s data readiness with another
    /// `CMSampleBuffer`'s data readiness.
    ///
    /// After calling this API, if `dataReadiness` is called, it will return
    /// `sampleBufferToTrack`'s data readiness. If `makeDataReady()` is
called, it
    /// will do it by making `sampleBufferToTrack` ready.
    ///
    /// Example of use: This allows bursting a multi-sample
`CMSampleBuffer` into
    /// single-sample `CMSampleBuffer`s before the data is ready. The
    /// single-sample `CMSampleBuffer`s will all track the multi-sample
    /// `CMSampleBuffer`'s data readiness.
    ///
    /// - Parameter sampleBufferToTrack: `CMSampleBuffer` being
tracked.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func trackDataReadiness _
CMSampleBuffer  throws

    /// Makes the sample buffer invalid, calling any installed invalidation
    /// handler.
    ///
    /// An invalid sample buffer cannot be used -- all accessors will throw
    /// `.invalidated`.
    ///
    /// It is not a good idea to do this to a sample buffer that another module
    /// may be accessing concurrently.
    ///
    /// Example of use: the invalidation handler could cancel pending I/O.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func invalidate   throws

    /// Sets the sample buffer's invalidation handler, which is called during
    /// `invalidate()`.
    ///
    /// A sample buffer can only have one invalidation handler.
    ///
    /// The invalidation handler is NOT called during ordinary sample buffer
    /// finalization.
    ///
    /// - Parameter body: Closure to be called during `invalidate()`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func setInvalidateHandler _       @escaping
 CMSampleBuffer  throws    Void  throws

    /// Queries whether a sample buffer is still valid.
    ///
    /// `false` if `invalidate()` was called, `true` otherwise.
    ///
    /// Does not perform any kind of exhaustive validation of the sample buffer.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var isValid  Bool   get

    /// The number of media samples in the `CMSampleBuffer`. `0` is
returned if
    /// there is an error.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var numSamples  Int   get

    /// The duration of the `CMSampleBuffer`. `CMTime.invalid` is
returned if
    /// there is an error.
    ///
    /// If the buffer contains out-of-presentation-order samples, any gaps in the
    /// presentation timeline are not represented in the returned duration.
    ///
    /// The returned duration is simply the sum of all the individual sample
    /// durations.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var duration  CMTime   get

    /// Numerically earliest sample presentation timestamp in the
`CMSampleBuffer`.
    ///
    /// `CMTime.invalid` is returned if there is an error.
    ///
    /// For in-presentation-order samples, this is the presentation timestamp of
    /// the first sample.
    ///
    ///  For out-of-presentation-order samples, this is the presentation timestamp
    ///  of the sample that will be presented first, which is not necessarily the
    ///  first sample in the buffer.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var presentationTimeStamp  CMTime   get

    ///  Numerically earliest sample decode timestamp in the
`CMSampleBuffer`.
    ///
    ///  `CMTime.invalid` is returned if there is an error.
    ///
    ///  The returned decode timestamp is always the decode timestamp of the
first
    ///  sample in the buffer, since even out-of-presentation-order samples are
    ///  expected to be in decode order in the buffer.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var decodeTimeStamp  CMTime   get

    ///  The output duration of the `CMSampleBuffer`.
    ///
    ///  `CMTime.invalid` is returned if there is an error.
    ///
    ///  The `outputDuration` is the duration minus any trimmed duration, all
    ///  divided by the `SpeedMultiplier`:
    ///  `(Duration - TrimDurationAtStart -
TrimDurationAtEnd) / SpeedMultiplier`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var outputDuration  CMTime   get

    ///  The output presentation timestamp of the `CMSampleBuffer`.
    ///
    ///  See `CMSampleBufferGetOutputPresentationTimeStamp`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var outputPresentationTimeStamp  CMTime   get

    ///  Sets an output presentation timestamp to be used in place of a calculated
    ///  value.
    ///
    ///  The output presentation timestamp is the time at which the decoded,
    ///  trimmed, stretched and possibly reversed samples should commence
being
    ///  presented. By default, this is calculated by
`outputPresentationTimeStamp`.
    ///
    ///  Call `setOutputPresentationTimeStamp` to explicitly set the value
for
```

```
    /// `outputPresentationTimeStamp` to return.
    ///
    /// For general forward playback in a scaled edit, the
    /// `OutputPresentationTimeStamp` should be set to:
    /// `((PresentationTimeStamp + TrimDurationAtStart -
EditStartMediaTime) / EditSpeedMultiplier) +
EditStartTrackTime`.
    /// For general reversed playback:
    /// `((PresentationTimeStamp + Duration -
TrimDurationAtEnd - EditStartMediaTime) / EditSpeedMultiplier)
+ EditStartTrackTime`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func setOutputPresentationTimeStamp _      CMTime
throws

    /// The output decode timestamp of the `CMSampleBuffer`.
    ///
    /// For consistency with `outputPresentationTimeStamp`, this is
calculated as:
    /// `OutputPresentationTimeStamp + ((DecodeTimeStamp -
PresentationTimeStamp) / SpeedMultiplier)`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var outputDecodeTimeStamp  CMTime   get

    /// Returns an array of `CMSampleTimingInfo` structs, one for each
sample in a
    /// `CMSampleBuffer`.
    ///
    /// If only one `CMSampleTimingInfo` struct is returned, it applies to all
    /// samples in the buffer.
    ///
    /// See documentation of `CMSampleTimingInfo` for details of how a
single
    /// `CMSampleTimingInfo` struct can apply to multiple samples.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func sampleTimingInfos   throws
 CMSampleTimingInfo

    /// Returns an array of output `CMSampleTimingInfo` structs, one for
each
    /// sample in a `CMSampleBuffer`.
    ///
    /// If only one `CMSampleTimingInfo` struct is returned, it applies to all
    /// samples in the buffer.
    /// See documentation of `CMSampleTimingInfo` for details of how a
single
    /// `CMSampleTimingInfo` struct can apply to multiple samples.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func outputSampleTimingInfos   throws
 CMSampleTimingInfo

    /// Returns a `CMSampleTimingInfo` struct describing a specified sample
in a
    /// `CMSampleBuffer`.
    ///
    /// A sample-specific `CMSampleTimingInfo` struct will be returned (ie.
with a
    /// sample-specific `presentationTimeStamp` and
`decodeTimeStamp`), even if a
    /// single `CMSampleTimingInfo` struct was used during creation to
describe
    /// all the samples in the buffer. If the sample index is not in the range
    /// `0..<numSamples`, `.sampleIndexOutOfRange` will be thrown.
    ///
    /// If there is no timingInfo in this `CMSampleBuffer`,
    /// `.bufferHasNoSampleTimingInfo` will be thrown.
    ///
    /// - Parameter sampleIndex: Sample index (`0` is first sample in
sbuf).
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func sampleTimingInfo                  CMItemIndex
throws    CMSampleTimingInfo

    /// Returns an array of sample sizes, one for each sample in a
`CMSampleBuffer`.
    ///
    /// If only one size entry is returned, all samples in the buffer are of this
    /// size.
    ///
    /// If there are no sample sizes in this `CMSampleBuffer`, an empty array
will
    /// be returned. This will be `true`, for example, if the samples in the
    /// buffer are non-contiguous (eg. non-interleaved audio, where the channel
    /// values for a single sample are scattered through the buffer), or if this
    /// `CMSampleBuffer` contains a `CVImageBuffer`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func sampleSizes   throws     Int

    /// Returns the size in bytes of a specified sample.
    ///
    /// - Parameter sampleIndex: Sample index (`0` is first sample in
sbuf).
    /// - Returns: Size in bytes of the specified sample in the
`CMSampleBuffer`.
```

```
    ///     If the sample index is not in the range `0..<numSamples`, a size of
`0`
    ///     will be returned. If there are no sample sizes in this
`CMSampleBuffer`,
    ///     a size of `0` will be returned. This will be true, for example, if the
    ///     samples in the buffer are non-contiguous (eg. non-interleaved audio,
    ///     where the channel values for a single sample are scattered through the
    ///     buffer), or if this `CMSampleBuffer` contains a `CVImageBuffer`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func sampleSize          Int     Int

    /// Total size in bytes of sample data in the `CMSampleBuffer`. If there are
    /// no sample sizes in this `CMSampleBuffer`, a size of `0` will be
returned.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var totalSampleSize  Int   get

    /// The format description of the samples in the `CMSampleBuffer`.
    /// `nil` is returned if there is an error.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var formatDescription  CMFormatDescription    get

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct PerSampleAttachmentsDictionary   Sequence

        /// The following keys may be attached to individual samples via
subscript
        @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS
6.0  visionOS 1.0
        public struct Key   @unchecked Sendable

            /// The raw type that can be used to represent all values of the
conforming
            /// type.
            ///
            /// Every distinct value of the conforming type has a
corresponding unique
            /// value of the `RawValue` type, but there may be values of the
`RawValue`
            /// type that don't have a corresponding value of the conforming
type.
            public typealias RawValue   CFString

            /// The corresponding value of the raw type.
            ///
            /// A new instance initialized with `rawValue` will be equivalent
```

to this

```
            /// instance. For example:
            ///
            ///     enum PaperSize: String {
            ///         case A4, A5, Letter, Legal
            ///     }
            ///
            ///     let selectedSize = PaperSize.Letter
            ///     print(selectedSize.rawValue)
            ///     // Prints "Letter"
            ///
            ///     print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue)!)
            ///     // Prints "true"
            public var rawValue  CFString

            /// Creates a new instance with the specified raw value.
            ///
            /// If there is no value of the type that corresponds with the
specified raw
            /// value, this initializer returns `nil`. For example:
            ///
            ///     enum PaperSize: String {
            ///         case A4, A5, Letter, Legal
            ///     }
            ///
            ///     print(PaperSize(rawValue: "Legal"))
            ///     // Prints "Optional("PaperSize.Legal")"
            ///
            ///     print(PaperSize(rawValue: "Tabloid"))
            ///     // Prints "nil"
            ///
            /// - Parameter rawValue: The raw value to use for the new
instance.
            public init           CFString

            /// Boolean (absence of this key implies Sync)
            public static let notSync
CMSampleBuffer PerSampleAttachmentsDictionary Key

            /// Boolean (absence of this key implies not Partial Sync. If
`notSync` is
            /// `false`, `partialSync` should be ignored.)
            public static let partialSync
CMSampleBuffer PerSampleAttachmentsDictionary Key

            /// `true`, `false`, or absent if unknown
            public static let hasRedundantCoding
CMSampleBuffer PerSampleAttachmentsDictionary Key
```

```
            /// `true`, `false`, or absent if unknown
            ///
            /// A frame is considered droppable if and only if
`isDependedOnByOthers`
            /// is present and set to `false`.
            public static let isDependedOnByOthers
CMSampleBuffer PerSampleAttachmentsDictionary Key

            /// `true` (e.g., non-I-frame), `false` (e.g. I-frame), or absent
if
            /// unknown
            public static let dependsOnOthers
CMSampleBuffer PerSampleAttachmentsDictionary Key

            public static let earlierDisplayTimesAllowed
CMSampleBuffer PerSampleAttachmentsDictionary Key

            public static let displayImmediately
CMSampleBuffer PerSampleAttachmentsDictionary Key

            public static let doNotDisplay
CMSampleBuffer PerSampleAttachmentsDictionary Key

            /// Indicates a video frame's level within a hierarchical frame
dependency
            /// structure.
            ///
            /// When present, the temporal level attachments among a group
of video
            /// frames provide information about where inter-frame
dependencies may
            /// and may not exist.
            ///
            /// The temporal level attachment, if present, is a positive number,
and
            /// indicates that this video frame does not depend on any video
frame
            /// with a greater temporal level.
            ///
            /// The attachment may be absent if no such information is
available.
            ///
            /// Corresponds to `'tscl'` sample group.
            public static let hevcTemporalLevelInfo
CMSampleBuffer PerSampleAttachmentsDictionary Key

            /// Boolean, optional. Corresponds to `'tsas'` sample group.
            public static let hevcTemporalSubLayerAccess
CMSampleBuffer PerSampleAttachmentsDictionary Key
```

```
            /// Boolean, optional. Corresponds to 'stsa' sample group.
            public static let
hevcStepwiseTemporalSubLayerAccess
CMSampleBuffer PerSampleAttachmentsDictionary Key

            /// Number, optional. Corresponds to `'sync'` sample group.
            public static let hevcSyncSampleNALUnitType
CMSampleBuffer PerSampleAttachmentsDictionary Key

            /// The audioIndependentSampleDecoderRefreshCount sample
attachment is
            /// only present if the audio sample is an IndependentFrame (IF,
value is
            /// non-zero) or ImmediatePlayoutFrame (IPF, value is zero).
            public static let
audioIndependentSampleDecoderRefreshCount
CMSampleBuffer PerSampleAttachmentsDictionary Key


        /// A type representing the sequence's elements.
        public typealias Element
CMSampleBuffer PerSampleAttachmentsDictionary Key          Any

        /// A type that provides the sequence's iteration interface and
        /// encapsulates its iteration state.
        public struct Iterator   IteratorProtocol

            /// Advances to the next element and returns it, or `nil` if no
next element
            /// exists.
            ///
            /// Repeatedly calling this method returns, in order, all the
elements of the
            /// underlying sequence. As soon as the sequence has run out of
elements, all
            /// subsequent calls return `nil`.
            ///
            /// You must not call this method if any other copy of this iterator
has been
            /// advanced with a call to its `next()` method.
            ///
            /// The following example shows how an iterator can be used
explicitly to
            /// emulate a `for`-`in` loop. First, retrieve a sequence's
iterator, and
            /// then call the iterator's `next()` method until it returns
`nil`.
            ///
            ///     let numbers = [2, 3, 5, 7]
            ///     var numbersIterator =
numbers.makeIterator()
            ///
            ///     while let num = numbersIterator.next() {
            ///         print(num)
            ///     }
            ///     // Prints "2"
            ///     // Prints "3"
            ///     // Prints "5"
            ///     // Prints "7"
            ///
            /// - Returns: The next element in the underlying sequence, if
a next element
            ///   exists; otherwise, `nil`.
            public mutating func next
CMSampleBuffer PerSampleAttachmentsDictionary Key
Any

            /// The type of element traversed by the iterator.
            @available iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0  macOS 10.15
            public typealias Element
CMSampleBuffer PerSampleAttachmentsDictionary Key        Any


        /// Returns an iterator over the elements of this sequence.
        public func makeIterator
CMSampleBuffer PerSampleAttachmentsDictionary Iterator

        public subscript
CMSampleBuffer PerSampleAttachmentsDictionary Key     Any


    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct SampleAttachmentsArray   Collection

        /// Returns the position immediately after the given index.
        ///
        /// The successor of an index must be well defined. For an index `i`
into a
        /// collection `c`, calling `c.index(after: i)` returns the same
index every
        /// time.
        ///
        /// - Parameter i: A valid index of the collection. `i` must be less
than
        ///   `endIndex`.
        /// - Returns: The index value immediately after `i`.
        public func index           Int      Int
        /// The position of the first element in a nonempty collection.
        ///
        /// If the collection is empty, `startIndex` is equal to `endIndex`.
        public var startIndex  Int

        /// The collection's "past the end" position---that is, the position one
        /// greater than the last valid subscript argument.
        ///
        /// When you need a range that includes the last element of a
collection, use
        /// the half-open range operator (`..<`) with `endIndex`. The
`..<` operator
        /// creates a range that doesn't include the upper bound, so it's always
        /// safe to use with `endIndex`. For example:
        ///
        ///     let numbers = [10, 20, 30, 40, 50]
        ///     if let index = numbers.firstIndex(of: 30) {
        ///         print(numbers[index ..< numbers.endIndex])
        ///     }
        ///     // Prints "[30, 40, 50]"
        ///
        /// If the collection is empty, `endIndex` is equal to `startIndex`.
        public var endIndex  Int

        /// A type that represents a position in the collection.
        ///
        /// Valid indices consist of the position of every element and a
        /// "past the end" position that's not valid for use as a subscript
        /// argument.
        public typealias Index   Int

        /// Accesses the element at the specified position.
        ///
        /// The following example accesses an element of an array through its
        /// subscript to print its value:
        ///
        ///     var streets = ["Adams", "Bryant", "Channing",
"Douglas", "Evarts"]
        ///     print(streets[1])
        ///     // Prints "Bryant"
        ///
        /// You can subscript a collection with any valid index other than the
        /// collection's end index. The end index refers to the position one past
        /// the last element of a collection, so it doesn't correspond with an
        /// element.
        ///
        /// - Parameter position: The position of the element to access.
`position`
        ///   must be a valid index of the collection that is not equal to the
        ///    `endIndex` property.
        ///
        /// - Complexity: O(1)
        public subscript         Int
CMSampleBuffer PerSampleAttachmentsDictionary

        /// A type representing the sequence's elements.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias Element
CMSampleBuffer PerSampleAttachmentsDictionary

        /// A type that represents the indices that are valid for subscripting the
        /// collection, in ascending order.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias Indices
DefaultIndices CMSampleBuffer SampleAttachmentsArray

        /// A type that provides the collection's iteration interface and
        /// encapsulates its iteration state.
        ///
        /// By default, a collection conforms to the `Sequence` protocol by
        /// supplying `IndexingIterator` as its associated `Iterator`
        /// type.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias Iterator
IndexingIterator CMSampleBuffer SampleAttachmentsArray

        /// A collection representing a contiguous subrange of this collection's
        /// elements. The subsequence shares indices with the original
collection.
        ///
        /// The default subsequence type for collections that don't define their
own
        /// is `Slice`.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias SubSequence
Slice CMSampleBuffer SampleAttachmentsArray


    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var sampleAttachments
CMSampleBuffer SampleAttachmentsArray


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
```

```
visionOS 1.0
extension CMSampleBuffer

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct SingleSampleBuffers   Sequence

        /// A type representing the sequence's elements.
        public typealias Element   CMSampleBuffer

        /// A type that provides the sequence's iteration interface and
        /// encapsulates its iteration state.
        public struct Iterator   IteratorProtocol

            /// Advances to the next element and returns it, or `nil` if no
next element
            /// exists.
            ///
            /// Repeatedly calling this method returns, in order, all the
elements of the
            /// underlying sequence. As soon as the sequence has run out of
elements, all
            /// subsequent calls return `nil`.
            ///
            /// You must not call this method if any other copy of this iterator
has been
            /// advanced with a call to its `next()` method.
            ///
            /// The following example shows how an iterator can be used
explicitly to
            /// emulate a `for`-`in` loop. First, retrieve a sequence's
iterator, and
            /// then call the iterator's `next()` method until it returns
`nil`.
            ///
            ///     let numbers = [2, 3, 5, 7]
            ///     var numbersIterator =
numbers.makeIterator()
            ///
            ///     while let num = numbersIterator.next() {
            ///         print(num)
            ///     }
            ///     // Prints "2"
            ///     // Prints "3"
            ///     // Prints "5"
            ///     // Prints "7"
            ///
            /// - Returns: The next element in the underlying sequence, if
a next element
            ///     exists; otherwise, `nil`.
            public mutating func next      CMSampleBuffer

            /// The type of element traversed by the iterator.
            @available iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0  macOS 10.15
            public typealias Element   CMSampleBuffer


        /// Returns an iterator over the elements of this sequence.
        public func makeIterator
CMSampleBuffer SingleSampleBuffers Iterator


    /// Get every individual sample in a sample buffer.
    ///
    /// Temporary sample buffers will be created for individual samples, referring
    /// to the sample data and containing its timing, size and attachments.
    /// If there are no sample sizes in the provided sample buffer,
kCMSampleBufferError_CannotSubdivide will be thrown. This will happen,
    /// for example, if the samples in the buffer are non-contiguous (eg.
    /// non-interleaved audio, where the channel values for a single sample are
    /// scattered through the buffer).
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func singleSampleBuffers   throws
CMSampleBuffer SingleSampleBuffers


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMSampleBuffer

    /// The following keys may be attached to sample buffers using
CMAttachmentBearerProtocol `.attachments`:
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct AttachmentKey   @unchecked Sendable

        /// The raw type that can be used to represent all values of the
conforming
        /// type.
        ///
        /// Every distinct value of the conforming type has a corresponding
unique
        /// value of the `RawValue` type, but there may be values of the
`RawValue`
        /// type that don't have a corresponding value of the conforming type.
        public typealias RawValue   CFString

        /// The corresponding value of the raw type.
        ///
        /// A new instance initialized with `rawValue` will be equivalent to this
        /// instance. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter, Legal
        ///     }
        ///
        ///     let selectedSize = PaperSize.Letter
        ///     print(selectedSize.rawValue)
        ///     // Prints "Letter"
        ///
        ///     print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue)!)
        ///     // Prints "true"
        public var rawValue  CFString

        /// Creates a new instance with the specified raw value.
        ///
        /// If there is no value of the type that corresponds with the specified
raw
        /// value, this initializer returns `nil`. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter, Legal
        ///     }
        ///
        ///     print(PaperSize(rawValue: "Legal"))
        ///     // Prints "Optional("PaperSize.Legal")"
        ///
        ///     print(PaperSize(rawValue: "Tabloid"))
        ///     // Prints "nil"
        ///
        /// - Parameter rawValue: The raw value to use for the new
instance.
        public init           CFString

        public static let resetDecoderBeforeDecoding
CMSampleBuffer AttachmentKey

        public static let drainAfterDecoding
CMSampleBuffer AttachmentKey

        public static let postNotificationWhenConsumed
CMSampleBuffer AttachmentKey

        public static let resumeOutput
CMSampleBuffer AttachmentKey

        /// Marks a transition from one source of buffers (eg. song) to another.
        ///
        /// For example, during gapless playback of a list of songs, this
attachment
        /// marks the first buffer from the next song. If this attachment is on a
        /// buffer containing no samples, the first following buffer that contains
        /// samples is the buffer that contains the first samples from the next
        /// song. This transition identifier should be unique within a playlist, so
        /// each transition in a playlist is uniquely identifiable. A counter that
        /// increments with each transition is a simple example.
        public static let transitionID
CMSampleBuffer AttachmentKey

        /// The duration that should be removed at the beginning of the sample
        /// buffer, after decoding.
        ///
        /// If this attachment is not present, the trim duration is zero (nothing
        /// removed). This is a `CMTime` in dictionary format as made by
        /// `CMTimeCopyAsDictionary`; use
`CMTimeMakeFromDictionary` to convert to
        /// `CMTime`. In cases where all the output after decoding the sample
buffer
        /// is to be discarded (e.g., the samples are only being decoded to
prime
        /// the decoder) the usual convention is to set
`trimDurationAtStart` to
        /// the whole duration and not to set a `trimDurationAtEnd`
attachment.
        ///
        /// Note that setting or removing `trimDurationAtStart` from a
sample buffer
        /// will not adjust an explicitly-set OutputPresentationTimeStamp.
        public static let trimDurationAtStart
CMSampleBuffer AttachmentKey

        /// The duration that should be removed at the end of the sample buffer,
        /// after decoding.
        ///
        /// If this attachment is not present, the trim duration is zero (nothing
        /// removed).
        ///
        /// This is a `CMTime` in dictionary format as made by
        /// `CMTimeCopyAsDictionary`; use
`CMTimeMakeFromDictionary` to convert to
        /// `CMTime`.
        public static let trimDurationAtEnd
CMSampleBuffer AttachmentKey

        /// The factor by which the sample buffer's presentation should be
        /// accelerated (eg, in a scaled edit).
        ///
        /// For normal playback the speed multiplier would be `1.0` (which is
used
        /// if this attachment is not present); for double-speed playback the
speed
        /// multiplier would be `2.0`, which would halve the output duration.
        /// Speed-multiplication factors take effect after trimming; see
        /// `outputDuration`. Note that this attachment principally provides
        /// information about the duration-stretching effect: by default, it should
        /// be implemented by rate conversion, but other attachments may
specify
        /// richer stretching operations -- for example, scaling without pitch
        /// shift, or pitch shift without changing duration. Sequences of
        /// speed-multiplied sample buffers should have explicit
        /// OutputPresentationTimeStamp attachments to clarify when each
should be
        /// output.
        public static let speedMultiplier
CMSampleBuffer AttachmentKey

        /// Indicates that the decoded contents of the sample buffer should be
        /// reversed.
        ///
        /// If this attachment is not present, the sample buffer should be played
        /// forwards as usual. Reversal occurs after trimming and speed
multipliers.
        public static let reverse
CMSampleBuffer AttachmentKey

        /// Fill the difference between discontiguous sample buffers with
silence.
        ///
        /// If a sample buffer enters a buffer queue and the presentation time
stamp
        /// between the previous buffer and the buffer with this attachment are
        /// discontiguous, handle the discontinuity by generating silence for the
        /// time difference.
        public static let fillDiscontinuitiesWithSilence
CMSampleBuffer AttachmentKey

        /// Marks an intentionally empty interval in the sequence of samples.
        ///
        /// The sample buffer's output presentation timestamp indicates when
the
        /// empty interval begins. Marker sample buffers with this attachment
are
        /// used to announce the arrival of empty edits.
        public static let emptyMedia
CMSampleBuffer AttachmentKey

        /// Marks the end of the sequence of samples.
        ///
        ///  Marker sample buffers with this attachment in addition to
`emptyMedia`
        ///  are used to indicate that no further samples are expected.
        public static let permanentEmptyMedia
CMSampleBuffer AttachmentKey

        ///  Tells that the empty marker should be dequeued immediately
regardless of
        ///  its timestamp.
        ///
        ///  Marker sample buffers with this attachment in addition to
`emptyMedia`
        ///  are used to tell that the empty sample buffer should be dequeued
        ///  immediately regardless of its timestamp. This attachment should
only be
        ///  used with sample buffers with the `emptyMedia` attachment.
        public static let displayEmptyMediaImmediately
CMSampleBuffer AttachmentKey

        ///  Indicates that sample buffer's decode timestamp may be used to
define
        ///  the previous sample buffer's duration.
        ///
        ///  Marker sample buffers with this attachment may be used in
situations
        ///  where sample buffers are transmitted before their duration is known.
In
        ///  such situations, normally the recipient may use each sample buffer's
        ///  timestamp to calculate the duration of the previous sample buffer.
The
        ///  marker sample buffer with this attachment is sent to provide the
        ///  timestamp for calculating the final sample buffer's duration.
        public static let endsPreviousSampleDuration
CMSampleBuffer AttachmentKey

        ///  Indicates the URL where the sample data is.
        ///
        ///  This key is only used for CMSampleBuffers representing sample
        ///  references. Such CMSampleBuffers:
        ///  -  have dataBuffer == nil and imageBuffer == nil
        ///  -  have dataReady == true and no makeDataReadyHandler
        ///  -  have a non-nil formatDescription
        ///  -  have numSamples > 0
        ///  -  have numSampleTimingEntries > 0 and numSampleSizeEntries >
0
        public static let sampleReferenceURL
CMSampleBuffer AttachmentKey

        ///  Indicates the byte offset at which the sample data begins.
        ///
        ///  This key is only used for CMSampleBuffers representing sample
        ///  references. Such CMSampleBuffers:
        ///  –  have dataBuffer == nil and imageBuffer == nil
        ///  –  have dataReady == true and no makeDataReadyHandler
        ///  –  have a non-nil formatDescription
        ///  –  have numSamples > 0
        ///  –  have numSampleTimingEntries > 0 and numSampleSizeEntries >
0
        public static let sampleReferenceByteOffset
CMSampleBuffer AttachmentKey

        ///  Indicates the decoder refresh count.
        ///
        ///  Sample buffers with this attachment may be used to identify the
audio
        ///  decoder refresh count.
        public static let gradualDecoderRefresh
CMSampleBuffer AttachmentKey

        ///  Indicates the reason the current video frame was dropped.
        ///
        ///  Sample buffers with this attachment contain no image or data buffer.
        ///  They mark a dropped video frame. This attachment identifies the
reason
        ///  for the droppage.
        public static let droppedFrameReason
CMSampleBuffer AttachmentKey

        ///  Indicates additional information regarding the dropped video frame.
        ///
        ///  Sample buffers with this attachment contain no image or data buffer.
        ///  They mark a dropped video frame. If present, this attachment
provides
        ///  additional information about the `droppedFrameReason`.
        public static let droppedFrameReasonInfo
CMSampleBuffer AttachmentKey

        ///  Indicates information about the lens stabilization applied to the
        ///  current still image buffer.
        ///
        ///  Sample buffers that have been captured with a lens stabilization
module
        ///  may have an attachment of
`stillImageLensStabilizationInfo` which has
        ///  information about the stabilization status during the capture. This
key
        ///  will not be present in `CMSampleBuffer`s coming from cameras
without a
        ///  lens stabilization module.
        public static let stillImageLensStabilizationInfo
```

```
CMSampleBuffer AttachmentKey

        /// Indicates the 3x3 camera intrinsic matrix applied to the current
sample
        /// buffer.
        ///
        /// Camera intrinsic matrix is a Data containing a matrix_float3x3, which
is
        /// column-major. It has the following contents:
        /// ```
        ///     fx 0  ox
        ///     0  fy oy
        ///     0  0  1
        /// ```
        /// `fx` and `fy` are the focal length in pixels. For square pixels, they
        /// will have the same value. `ox` and `oy` are the coordinates of the
        /// principal point. The origin is the upper left of the frame.
        public static let cameraIntrinsicMatrix
CMSampleBuffer AttachmentKey

        /// Indicates that the current or next video sample buffer should be
forced
        /// to be encoded as a key frame.
        ///
        /// A value of `true` for `forceKeyFrame` indicates that the current
or next
        /// video sample buffer processed in the stream should be forced to be
        /// encoded as a key frame. If this attachment is present and `true`
on a
        /// sample buffer with a video frame, that video frame will be forced to
        /// become a key frame. If the sample buffer for which this is present
and
        /// `true` does not have a valid video frame, the next sample buffer
        /// processed that contains a valid video frame will be encoded as a
key
        /// frame.
        ///
        /// Usual care should be taken when setting attachments on sample
buffers
        /// whose orgins and destinations are ambiguous. For example, seting
        /// attachments is not thread-safe, and `CMSampleBuffer`s may be
used in
        /// multiple sample buffer streams in a given system. This can lead to
        /// crashes during concurrent access and/or unexpected behavior on
alternate
        /// sample buffer streams. Therefore, unless the orgin and destination
of a
        /// sample buffer is known, the general recommended practice is to
        /// synthesize an empty sample buffer with this attachment alone and
insert
        /// it into the sample buffer stream ahead of the concrete sample buffer
```

```
        /// rather than setting this attachment on the concrete sample buffer
        /// itself.
        public static let forceKeyFrame
CMSampleBuffer AttachmentKey



@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMSampleBuffer

    /// Posted on a `CMSampleBuffer` by `setDataReadiness(.ready)`
when the buffer
    /// becomes ready.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static let dataBecameReady  NSNotification Name

    /// Posted on a `CMSampleBuffer` by
`setDataReadiness(.failed())` to report
    /// that the buffer will never become ready.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static let dataFailed  NSNotification Name

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct NotificationKey   @unchecked Sendable

        /// The raw type that can be used to represent all values of the
conforming
        /// type.
        ///
        /// Every distinct value of the conforming type has a corresponding
unique
        /// value of the `RawValue` type, but there may be values of the
`RawValue`
        /// type that don't have a corresponding value of the conforming type.
        public typealias RawValue   CFString

        /// The corresponding value of the raw type.
        ///
        /// A new instance initialized with `rawValue` will be equivalent to this
        /// instance. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter, Legal
        ///     }
        ///
        ///     let selectedSize = PaperSize.Letter
        ///     print(selectedSize.rawValue)
        ///     // Prints "Letter"
        ///
        ///     print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue)!)
        ///     // Prints "true"
        public var rawValue  CFString

        /// Creates a new instance with the specified raw value.
        ///
        /// If there is no value of the type that corresponds with the specified
raw
        /// value, this initializer returns `nil`. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter, Legal
        ///     }
        ///
        ///     print(PaperSize(rawValue: "Legal"))
        ///     // Prints "Optional("PaperSize.Legal")"
        ///
        ///     print(PaperSize(rawValue: "Tabloid"))
        ///     // Prints "nil"
        ///
        /// - Parameter rawValue: The raw value to use for the new
instance.
        public init           CFString

        /// Attached to `CMSampleBuffer.dataFailed`
        public static let status
CMSampleBuffer NotificationKey



@available macOS 14.0  iOS 17.0  tvOS 17.0  watchOS 10.0
visionOS 1.0
extension Sequence where Self Element   CMTag

    /**
      Filters a sequence of tags based on matching the specified category.  Returns
the tags that match the specified category.

      - Parameters:
      - category: The category to match.
      */
    public func filter T
CMTypedTag T  Category      CMTypedTag T   where T   Sendable

    /**
      Finds the first tag matching the specified category and returns the value of the
```

```
matching tag.

      - Parameters:
      - category:  The category to match.
      */
     public func firstValue T
CMTypedTag T  Category     T  where T   Sendable

     /**
      Finds and returns the first tag matching the specified category.

      - Parameters:
      - category:  The category to match.
      */
     public func first T
CMTypedTag T  Category     CMTypedTag T   where T   Sendable


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMBufferQueue

     ///  Handlers provided to `CMBufferQueue` initializers, for use by the
queue
     ///  in interrogating the buffers that it will see.
     @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
     public struct Handlers   @unchecked Sendable

         ///  This handler is called from `firstDecodeTimeStamp` (once),
and from
         ///  `minDecodeTimeStamp` (multiple times). It should return the
decode
         ///  timestamp of the buffer. If there are multiple samples in the buffer,
         ///  this handler should return the minimum decode timestamp in the
buffer.
         ///  Can be `nil` (`firstDecodeTimeStamp` and
`minDecodeTimeStamp` will
         ///  return `CMTime.invalid`).
         public let getDecodeTimeStamp  CMBufferGetTimeHandler

         ///  This handler is called from `firstPresentationTimeStamp`
(once) and from
         ///  `minPresentationTimeStamp` (multiple times). It should return
the
         ///  presentation timestamp of the buffer. If there are multiple samples in
         ///  the buffer, this handler should return the minimum presentation
         ///  timestamp in the buffer.
         ///  Can be `nil` (`firstPresentationTimeStamp` and
         ///  `minPresentationTimeStamp` will return
`CMTime.invalid`).
         public let getPresentationTimeStamp
CMBufferGetTimeHandler

         /// This handler is called (once) during enqueue and dequeue
operations to
         /// update the total duration of the queue.
         public let getDuration  CMBufferGetTimeHandler

         /// This handler is called from `dequeueIfDataReady()`, to ask if
the buffer
         /// that is about to be dequeued is ready.
         /// Can be `nil` (data will be assumed to be ready).
         public let isDataReady  CMBufferGetBooleanHandler

         /// This handler is called (multiple times) from `enqueue()`, to
perform an
         /// insertion sort. Can be `nil` (queue will be FIFO).
         public let compare  CMBufferCompareHandler

         /// If triggers of type `.whenDataBecomesReady` are installed, the
queue
         /// will listen for this notification on the head buffer.
         /// Can be `nil` (then the queue won't listen for it).
         public let dataBecameReadyNotification  String

         /// This handler is called (once) during enqueue and dequeue operation
to
         /// update the total size of the queue. Can be `nil`.
         public let getSize  CMBufferGetSizeHandler

         /// Builder helper.
         ///
         /// This builder let you create new handlers from an existing set of
         /// handlers:
         /// ```
         /// let handlers =
CMBufferQueue.Handlers.unsortedSampleBuffers.withHandlers {
         ///   $0.compare { lhs, rhs in
         ///     let lhs = lhs as! CMSampleBuffer
         ///     let rhs = rhs as! CMSampleBuffer
         ///     if lhs.duration == rhs.duration
{ return .compareEqualTo }
         ///     else if lhs.duration < rhs.duration
{ return .compareLessThan }
         ///     else { return .compareGreaterThan }
         ///    }
         ///  }
         /// ```
         public struct Builder   @unchecked Sendable
```

```
            public var dataBecameReadyNotification  String

            ///  Set the getDecodeTimeStamp handler
            public mutating func getDecodeTimeStamp _
@escaping CMBufferGetTimeHandler

            ///  Set the getPresentationTimeStamp handler
            public mutating func getPresentationTimeStamp _
      @escaping CMBufferGetTimeHandler

            ///  Set the getDuration handler
            public mutating func getDuration _       @escaping
CMBufferGetTimeHandler

            ///  Set the isDataReady handler
            public mutating func isDataReady _       @escaping
CMBufferGetBooleanHandler

            ///  Set the compare handler
            public mutating func compare _       @escaping
CMBufferCompareHandler

            ///  Set the getSize handler
            public mutating func getSize _       @escaping
CMBufferGetSizeHandler


        ///  Creates a `Handlers` using a `Builder`
        public init                     inout
CMBufferQueue Handlers Builder     Void

        ///  Creates a `Handlers` using a `Builder`, using `self` as
default values
        public func withHandlers _       inout
CMBufferQueue Handlers Builder     Void
CMBufferQueue Handlers

        ///  Callbacks for unsorted `CMSampleBuffer`s, provided as a
convenience.
        public static let unsortedSampleBuffers
CMBufferQueue Handlers

        ///  Callbacks for `CMSampleBuffer`s sorted by output presentation
timestamp,
        ///  provided as a convenience.
        public static let outputPTSSortedSampleBuffers
CMBufferQueue Handlers
```

```
@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMBufferQueue

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct Error   Sendable

        public static let allocationFailed  NSError

        public static let requiredParameterMissing  NSError

        public static let invalidCMBufferCallbacksStruct
NSError

        public static let enqueueAfterEndOfData  NSError

        public static let queueIsFull  NSError

        public static let badTriggerDuration  NSError

        public static let
cannotModifyQueueFromTriggerCallback  NSError

        public static let invalidTriggerCondition  NSError

        public static let invalidTriggerToken  NSError

        public static let invalidBuffer  NSError



@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMBufferQueue

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public typealias TriggerToken   CMBufferQueueTriggerToken

    ///  A condition to be associated with a TriggerToken.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public enum TriggerCondition   Sendable

        ///  Trigger fires when queue duration becomes `<` the specified
duration.
        case whenDurationBecomesLessThan CMTime
        /// Trigger fires when queue duration becomes `<=` the specified 
duration.
        case whenDurationBecomesLessThanOrEqualTo CMTime

        /// Trigger fires when queue duration becomes `>` the specified 
duration.
        case whenDurationBecomesGreaterThan CMTime

        /// Trigger fires when queue duration becomes `>=` the specified 
duration.
        case whenDurationBecomesGreaterThanOrEqualTo CMTime

        /// Trigger fires when minimum presentation timestamp changes.
        case whenMinPresentationTimeStampChanges

        /// Trigger fires when maximum presentation timestamp changes.
        case whenMaxPresentationTimeStampChanges

        /// Trigger fires when next dequeueable buffer becomes ready (ie,
        /// `dequeueIfDataReady()` will now succeed).
        case whenDataBecomesReady

        /// Trigger fires when `isAtEndOfData` becomes true.
        case whenEndOfDataReached

        /// Trigger fires when `reset()` is called.
        case whenReset

        /// Trigger fires when buffer count becomes `<` the specified threshold 
number.
        case whenBufferCountBecomesLessThan CMItemCount

        /// Trigger fires when buffer count becomes `>` the specified threshold 
number.
        case whenBufferCountBecomesGreaterThan CMItemCount


    /// Enqueues a buffer.
    ///
    /// The `buffer` is retained by the queue, so the client can safely release
    /// the buffer if it has no further use for it.
    ///
    /// If the compare handler is not `nil`, this API performs an insertion sort
    /// using that compare operation.
    ///
    /// If the validation handler is not `nil`, this API calls it; if it throws,
    /// the buffer will not be enqueued and this API will rethrow the error.
    ///
    /// - Parameter buffer: The buffer to enqueue.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func enqueue _        CMBuffer  throws

    /// Dequeues a buffer.
    ///
    /// - Returns: The dequeued buffer. Will be `nil` if the queue is empty.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func dequeue      CMBuffer

    /// Dequeues a buffer if it is ready.
    ///
    /// - Returns: The dequeued buffer. Will be `nil` if the queue is empty,
or if
    /// the buffer to be dequeued is not yet ready.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func dequeueIfDataReady      CMBuffer

    /// Retrieves the next-to-dequeue buffer but leaves it in the queue.
    ///
    /// Note that with non-FIFO queues it's not guaranteed that the next dequeue
    /// will return this particular buffer (if an intervening enqueue adds a
    /// buffer that will dequeue next).
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var head  CMBuffer    get

    /// Returns whether or not the `CMBufferQueue` is empty.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var isEmpty  Bool    get

    /// Marks the `CMBufferQueue` with EOD.
    ///
    /// All subsequent enqueues will be rejected until `reset()` is called.
    /// Subsequent dequeues will succeed as long as the queue is not empty.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func markEndOfData    throws

    /// Returns whether or not the `CMBufferQueue` has been marked with
EOD.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var containsEndOfData  Bool    get

    /// Returns whether or not the `CMBufferQueue` has been marked with
EOD, and
```

```
    /// is now empty.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var isAtEndOfData  Bool   get

    /// Resets the `CMBufferQueue`. Empties the queue, and clears any EOD
mark.
    ///
    /// All buffers in the queue are released. Triggers are not removed, however,
    /// and will be called appropriately as the queue duration goes to `.zero`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func reset   throws

    /// Calls a closure for every buffer in the queue, then resets the queue.
    ///
    /// - Parameter body:  Closure to be called for each buffer. The closure
should
    ///     not make other calls to the buffer queue.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func reset _         CMBuffer  throws         throws

    /// Gets the number of buffers in the queue.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var bufferCount  CMItemCount   get

    /// Gets the duration.
    ///
    /// The duration of the `CMBufferQueue` is the sum of all the individual
    /// buffer durations, as reported by the `getDuration` handler. If there are
no
    /// buffers in the queue, `CMTime.zero` will be returned.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var duration  CMTime   get

    /// Gets the earliest decode timestamp.
    ///
    /// The search for earliest decode timstamp is performed in this API.
    /// If you know your queue is in decode order, `firstDecodeTimeStamp`
is a
    /// faster alternative. If the `getDecodeTimeStamp` handler is `nil`,
    /// `CMTime.invalid` will be returned.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var minDecodeTimeStamp  CMTime   get

    /// Gets the decode timestamp of the first buffer.
    ///
    /// This API is is a faster alternative to `minDecodeTimeStamp`, but only
    /// gives the same answer if your queue is in decode order.
    ///
    /// If the `getDecodeTimeStamp` handler is `nil`, `CMTime.invalid`
will be
    /// returned.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var firstDecodeTimeStamp  CMTime   get

    /// Gets the earliest presentation timestamp.
    ///
    /// The search for earliest presentation timstamp is performed in this API. If
    /// you know your queue is sorted by presentation time,
    /// `firstPresentationTimeStamp` is a faster alternative. If the
    /// `getPresentationTimeStamp` handler is `nil`,
`CMTime.invalid` will be
    /// returned.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var minPresentationTimeStamp  CMTime   get

    /// Gets the presentation timestamp of the first buffer.
    ///
    /// This API is is a faster alternative to `minPresentationTimeStamp`,
but
    /// only  works if you know your queue is sorted by presentation timestamp. If
    /// the `getPresentationTimeStamp` handler is `nil`,
`CMTime.invalid` will be
    /// returned.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var firstPresentationTimeStamp  CMTime   get

    /// Gets the greatest presentation timestamp.
    ///
    /// If the `getPresentationTimeStamp` handler is `nil`,
`CMTime.invalid` will
    /// be returned.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var maxPresentationTimeStamp  CMTime   get

    /// Gets the greatest end presentation timestamp.
    ///
    /// This is the maximum end time (PTS + duration) of buffers in the queue.
    /// If the `getPresentationTimeStamp` handler is `nil`,
`CMTime.invalid` will
    /// be returned.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var endPresentationTimeStamp  CMTime   get

    /// Gets the total size.
    ///
    /// The total size of the `CMBufferQueue` is the sum of all the individual
    /// buffer sizes, as reported by the `getTotalSize` handler. If there are no
    /// buffers in the queue, `0` will be returned.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var totalSize  Int   get

    /// Installs a trigger.
    ///
    /// The returned trigger token can be passed to `testTrigger` and
    /// `removeTrigger`.
    ///
    /// The returned trigger can be discarded (client doesn't need to test or
    /// remove trigger), and the body parameter can be `nil` (client doesn't
need
    /// callbacks, but rather will explicitly test the trigger). One of these two
    /// parameters must be non-`nil`, however, since an untestable trigger that
    /// does not perform a callback is meaningless. If the trigger condition is
    /// already true, `installTrigger` will call the `body`.
    ///
    /// - Parameters:
    ///   - condition:  The condition to be tested when evaluating the
trigger.
    ///   - body:  Closure to be called when the trigger condition becomes
true.
    ///       Can be `nil`, if client intends only to explicitly test the condition.
    /// - Returns:  Trigger token which can be used with `testTrigger`
and
    ///   `removeTrigger`. Can be discarded if client has no need to
explicitly
    ///   test or remove the trigger.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func installTrigger
CMBufferQueue TriggerCondition  _
CMBufferQueueTriggerHandler    nil  throws
CMBufferQueue TriggerToken

    /// Removes a previously installed trigger.
    ///
    /// Triggers will automatically be removed when a queue is finalized.
    /// However, if more than one module has access to a queue, it may be hard
    /// for an individual module to know when the queue is finalized since other
    /// modules may retain it. To address this concern, modules should remove
    /// their triggers before they themselves are finalized.
    ///
    /// - Parameter triggerToken: Trigger to remove from the queue
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func removeTrigger _
CMBufferQueue TriggerToken  throws

    /// Tests whether the trigger condition is true.
    ///
    /// Whereas the trigger callback will only be called when the condition goes
    /// from `false` to `true`, `testTrigger` always returns the
condition's
    /// current status.
    /// The `triggerToken` must be one that has been installed on this
queue.
    ///
    /// - Parameter triggerToken: Trigger to test.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func testTrigger _
CMBufferQueue TriggerToken     Bool

    /// A sequence of `CMBuffer`s.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct Buffers   Sequence

        /// A type representing the sequence's elements.
        public typealias

        /// A type that provides the sequence's iteration interface and
        /// encapsulates its iteration state.
        public struct Iterator   IteratorProtocol

            /// Advances to the next element and returns it, or `nil` if no
next element
            /// exists.
            ///
            /// Repeatedly calling this method returns, in order, all the
elements of the
            /// underlying sequence. As soon as the sequence has run out of
elements, all
            /// subsequent calls return `nil`.
            ///
            /// You must not call this method if any other copy of this iterator
has been
            /// advanced with a call to its `next()` method.
            ///
            /// The following example shows how an iterator can be used
```

```
explicitly to
            /// emulate a `for`-`in` loop. First, retrieve a sequence's
iterator, and
            /// then call the iterator's `next()` method until it returns
`nil`.
            ///
            ///     let numbers = [2, 3, 5, 7]
            ///     var numbersIterator =
numbers.makeIterator()
            ///
            ///     while let num = numbersIterator.next() {
            ///         print(num)
            ///     }
            ///     // Prints "2"
            ///     // Prints "3"
            ///     // Prints "5"
            ///     // Prints "7"
            ///
            /// - Returns: The next element in the underlying sequence, if
a next element
            ///   exists; otherwise, `nil`.
            public mutating func next      CMBuffer

            /// The type of element traversed by the iterator.
            @available iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0  macOS 10.15
            public typealias Element   AnyObject


        /// Returns an iterator over the elements of this sequence.
        public func makeIterator
CMBufferQueue Buffers Iterator


    /// Accesses buffers in a `CMBufferQueue`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var buffers  CMBufferQueue Buffers   get

    /// Sets a function that `enqueue` will call to validate buffers before adding
    /// them.
    ///
    /// `enqueue` will call this closure to validate buffers.
    ///
    /// Throw an error code if the buffer should be rejected; `enqueue` will
    /// throw this error to the caller.
    ///
    /// If you do not have a more descriptive error code, use
    /// `Error.invalidBuffer`.
    ///
    /// - Parameter body:  Closure that will validate each buffer enqueued.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func setValidationHandler _       @escaping
 CMBufferQueue  CMBuffer  throws    Void

    /// The `CFTypeID` corresponding to `CMBufferQueue`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public class var typeID  CFTypeID   get


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMFormatDescription

    /// Extensions are a collection of `Key`/`Value` pairs
    ///
    /// They can be created using the set of known keys, and the key-specific
    /// `Value` factory, or using the underlying representation of `CFString`
and
    /// `CFPropertyList`:
    ///
    ///     var extensions = CMFormatDescription.Extensions()
    ///     extensions[.cleanAperture] = .cleanAperture(
    ///       width: 320, height: 240, horizontalOffet: 10,
verticalOffset: 20)
    ///
    /// is equivalent to:
    ///
    ///     let extensions =
CMFormatDescription.Extensions(base:
    ///       [kCMFormatDescriptionExtension_CleanAperture: [
    ///         kCMFormatDescriptionKey_CleanApertureWidth:
320,
    ///         kCMFormatDescriptionKey_CleanApertureHeight:
240,
    ///
kCMFormatDescriptionKey_CleanApertureHorizontalOffset: 10,
    ///
kCMFormatDescriptionKey_CleanApertureVerticalOffset: 20,
    ///       ] as AnyObject])
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct Extensions   @unchecked Sendable

        @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS
6.0  visionOS 1.0
        public struct Key   @unchecked Sendable
```

```
            ///  The raw type that can be used to represent all values of the
conforming
            ///  type.
            ///
            ///  Every distinct value of the conforming type has a
corresponding unique
            ///  value of the `RawValue` type, but there may be values of the
`RawValue`
            ///  type that don't have a corresponding value of the conforming
type.
            public typealias RawValue   CFString

            ///  The corresponding value of the raw type.
            ///
            ///  A new instance initialized with `rawValue` will be equivalent
to this
            ///  instance. For example:
            ///
            ///     enum PaperSize: String {
            ///         case A4, A5, Letter, Legal
            ///     }
            ///
            ///     let selectedSize = PaperSize.Letter
            ///     print(selectedSize.rawValue)
            ///     // Prints "Letter"
            ///
            ///     print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue)!)
            ///     // Prints "true"
            public var rawValue  CFString

            ///  Creates a new `Key` backed by `rawValue`
            public init           CFString

            ///  This extension contains a media-type-specific dictionary of
settings
            ///  used to produce a compressed media buffer.
            public static let originalCompressionSettings
CMFormatDescription Extensions Key

            ///  Sample description extension atoms that were not translated
into other
            ///  entries in the extensions dictionary.
            public static let sampleDescriptionExtensionAtoms
CMFormatDescription Extensions Key

            ///  Preserves the original SampleDescription data.
            public static let verbatimSampleDescription
CMFormatDescription Extensions Key
```

```
                /// Preserves the original ISOSampleEntry data.
                public static let verbatimISOSampleEntry
CMFormatDescription Extensions Key

                /// String
                public static let formatName
CMFormatDescription Extensions Key

                /// Number with depth value as directed by
                ///
http://developer.apple.com/qa/qa2001/qa1183.html
                public static let depth
CMFormatDescription Extensions Key

                /// Use
`.cleanAperture(width:height:horizontalOffet:verticalOffset:)`
                public static let cleanAperture
CMFormatDescription Extensions Key

                /// Number, 1 or 2
                public static let fieldCount
CMFormatDescription Extensions Key

                /// One of `FieldDetail` values
                public static let fieldDetail
CMFormatDescription Extensions Key

                /// Use
`.pixelAspectRatio(horizontalSpacing:verticalSpacing:)`
                public static let pixelAspectRatio
CMFormatDescription Extensions Key

                /// Describes the color primaries. One of `ColorPrimaries`
values
                public static let colorPrimaries
CMFormatDescription Extensions Key

                /// Describes the transfer function. One of
`TransferFunction` values
                public static let transferFunction
CMFormatDescription Extensions Key

                /// Number describing the gamma level, used in absence of (or
ignorance
                /// of) `transferFunction`
                public static let gammaLevel
CMFormatDescription Extensions Key

                /// Describes the color matrix for YCbCr->RGB. One of
```

```
`YCbCrMatrix` values
            public static let yCbCrMatrix
CMFormatDescription Extensions Key

            /// Boolean; by default, `false` for YCbCr-based compressed
formats,
            /// indicating that pixel values are video-range rather than full-
range
            public static let fullRangeVideo
CMFormatDescription Extensions Key

            /// Data
            public static let iccProfile
CMFormatDescription Extensions Key

            /// Number describing the bytes per row of raster pixel data (not
used for
            /// compressed, planar, tiled or downsampled formats)
            public static let bytesPerRow
CMFormatDescription Extensions Key

            /// Chroma siting information. For progressive images, only the
TopField
            /// value is used. One of `ChromaLocation` values
            public static let chromaLocationTopField
CMFormatDescription Extensions Key

            /// Chroma siting information. For progressive images, only the
TopField
            /// value is used. One of `ChromaLocation` values
            public static let chromaLocationBottomField
CMFormatDescription Extensions Key

            /// One of `MPEG2VideoProfile` values
            public static let conformsToMPEG2VideoProfile
CMFormatDescription Extensions Key

            /// Number
            public static let temporalQuality
CMFormatDescription Extensions Key

            /// Number
            public static let spatialQuality
CMFormatDescription Extensions Key

            /// Number
            public static let version
CMFormatDescription Extensions Key

            /// Number
            public static let revisionLevel
CMFormatDescription Extensions Key

            /// String of fourCC
            public static let vendor
CMFormatDescription Extensions Key

            /// Data (24 bytes); big-endian structure; same as
            /// `kCVImageBufferMasteringDisplayColorVolumeKey`;
matches payload of
            /// ISO/IEC 23008-2:2015(E), D.2.28 Mastering display colour
volume SEI
            /// message
            public static let masteringDisplayColorVolume
CMFormatDescription Extensions Key

            /// Data(4 bytes); big-endian structure; same as
            /// `kCVImageBufferContentLightLevelInfoKey`
            public static let contentLightLevelInfo
CMFormatDescription Extensions Key

            /// String (usually `TransferFunction.itu_R_2100_HLG`
when used);
            /// corresponds to D.2.38 Alternative Transfer Characteristics SEI
message
            public static let
alternativeTransferCharacteristics
CMFormatDescription Extensions Key

            /// String (Auxiliary type URN)
            public static let auxiliaryTypeInfo
CMFormatDescription Extensions Key

            /// One of `AlphaChannelMode` values
            public static let alphaChannelMode
CMFormatDescription Extensions Key

            /// Boolean; used to signal the presence of alpha channel in the
bitstream
            public static let containsAlphaChannel
CMFormatDescription Extensions Key

            /// Use `textDisplayFlags(_:)`
            public static let displayFlags
CMFormatDescription Extensions Key

            /// Use `qtTextColor(red:green:blue:alpha:)` or
            /// `mobile3GPPTextColor(red:green:blue:alpha:)
            public static let backgroundColor
CMFormatDescription Extensions Key
```

```
            /// Use `textRect(top:left:bottom:right:)`
            public static let defaultTextBox
CMFormatDescription Extensions Key

            /// Use
`qtTextDefaultStyle(startChar:height:ascent:localFontID:fontFa
ce:fontSize:foregroundColor:defaultFontName:)`
            /// or
`mobile3GPPTextDefaultStyle(startChar:endChar:localFontID:font
Face:fontSize:foregroundColor:)
            public static let defaultStyle
CMFormatDescription Extensions Key

            /// Use `textJustification(_:)`. Specific to
`.mobile3GPP`
            public static let horizontalJustification
CMFormatDescription Extensions Key

            /// Use `textJustification(_:)`. Specific to
`.mobile3GPP`
            public static let verticalJustification
CMFormatDescription Extensions Key

            /// Use `fontTable(_:)`. Specific to `.mobile3GPP`
            public static let fontTable
CMFormatDescription Extensions Key

            /// Use `textJustification(_:)`. Specific to `.qt`
            public static let textJustification
CMFormatDescription Extensions Key

            /// String
            public static let defaultFontName
CMFormatDescription Extensions Key

            /// Use `sourceReferenceName(value:langCode:)`
            public static let sourceReferenceName
CMFormatDescription Extensions Key

            public static let metadataKeyTable
CMFormatDescription Extensions Key

            /// Data(8 bytes); big-endian structure; same as
kCVImageBufferAmbientViewingEnvironmentKey; matches payload of ISO/IEC
23008-2:2017, D.2.39 ambient viewing environment SEI message
            @available macOS 12.0  iOS 15.0  tvOS 15.0
watchOS 8.0  visionOS 1.0
            public static let ambientViewingEnvironment
CMFormatDescription Extensions Key
```

```
            ///  Number (such as 8, 10, 12, 16, etc).
            @available macOS 12.0  iOS 15.0  tvOS 15.0
watchOS 8.0  visionOS 1.0
            public static let bitsPerComponent
CMFormatDescription Extensions Key


        public struct Value   @unchecked Sendable

            ///  The underlying representation of a `Value`
            public var propertyListRepresentation
CFPropertyList

            ///  Creates a `Value` from a `CFPropertyList`.
            public init _        CFPropertyList

            public static func number T  _        T
CMFormatDescription Extensions Value where T   Numeric

            public static func string _        String
CMFormatDescription Extensions Value

            public static func string _        CFString
CMFormatDescription Extensions Value

            @available macOS 12.0  iOS 15.0  tvOS 15.0
watchOS 8.0  visionOS 1.0
            public static func data _        CFData
CMFormatDescription Extensions Value

            ///  Used for `.cleanAperture`
            public static func cleanAperture Width  Height
Horizontal  Vertical        Width          Height
                 Horizontal                  Vertical
CMFormatDescription Extensions Value where Width   Numeric
Height   Numeric  Horizontal   Numeric  Vertical   Numeric

            ///  Used for `.cleanAperture`.
            ///
            ///  Some modules only read the float-valued keys, so both
the ...Rational
            ///  keys and the corresponding floating-point keys are set.
            public static func cleanAperture
            Int                 Int                         Int
             Int                                    Int
             Int                                   Int
             Int      CMFormatDescription Extensions Value
        public struct FieldDetail   @unchecked Sendable

            /// The raw type that can be used to represent all values of
the conforming
            /// type.
            ///
            /// Every distinct value of the conforming type has a
corresponding unique
            /// value of the `RawValue` type, but there may be values
of the `RawValue`
            /// type that don't have a corresponding value of the
conforming type.
            public typealias RawValue   CFString

            /// The corresponding value of the raw type.
            ///
            /// A new instance initialized with `rawValue` will be
equivalent to this
            /// instance. For example:
            ///
            ///     enum PaperSize: String {
            ///         case A4, A5, Letter, Legal
            ///     }
            ///
            ///     let selectedSize = PaperSize.Letter
            ///     print(selectedSize.rawValue)
            ///     // Prints "Letter"
            ///
            ///     print(selectedSize ==
PaperSize(rawValue: selectedSize.rawValue)!)
            ///     // Prints "true"
            public var rawValue  CFString

            /// Creates a new instance with the specified raw value.
            ///
            /// If there is no value of the type that corresponds with the
specified raw
            /// value, this initializer returns `nil`. For example:
            ///
            ///     enum PaperSize: String {
            ///         case A4, A5, Letter, Legal
            ///     }
            ///
            ///     print(PaperSize(rawValue: "Legal"))
            ///     // Prints
"Optional("PaperSize.Legal")"
            ///
            ///     print(PaperSize(rawValue: "Tabloid"))
            ///     // Prints "nil"
            ///
```

```
                /// - Parameter rawValue:  The raw value to use for
the new instance.
                public init           CFString

                public static let temporalTopFirst
CMFormatDescription Extensions Value FieldDetail

                public static let temporalBottomFirst
CMFormatDescription Extensions Value FieldDetail

                public static let spatialFirstLineEarly
CMFormatDescription Extensions Value FieldDetail

                public static let spatialFirstLineLate
CMFormatDescription Extensions Value FieldDetail


            /// Used for `.fieldDetail`
            public static func fieldDetail _
CMFormatDescription Extensions Value FieldDetail
CMFormatDescription Extensions Value

            /// Used for `.pixelAspectRatio`
            public static func pixelAspectRatio Horizontal
Vertical                     Horizontal
Vertical     CMFormatDescription Extensions Value where
Horizontal   Numeric   Vertical   Numeric

            public struct ColorPrimaries   @unchecked Sendable


                /// The raw type that can be used to represent all values of
the conforming
                /// type.
                ///
                /// Every distinct value of the conforming type has a
corresponding unique
                /// value of the `RawValue` type, but there may be values
of the `RawValue`
                /// type that don't have a corresponding value of the
conforming type.
                public typealias RawValue   CFString

                /// The corresponding value of the raw type.
                ///
                /// A new instance initialized with `rawValue` will be
equivalent to this
                /// instance. For example:
                ///
                ///     enum PaperSize: String {
```

```
                ///         case A4, A5, Letter, Legal
                ///     }
                ///
                ///     let selectedSize = PaperSize.Letter
                ///     print(selectedSize.rawValue)
                ///     // Prints "Letter"
                ///
                ///     print(selectedSize ==
PaperSize(rawValue: selectedSize.rawValue)!)
                ///     // Prints "true"
                public var rawValue  CFString

                /// Creates a new instance with the specified raw value.
                ///
                /// If there is no value of the type that corresponds with the
specified raw
                /// value, this initializer returns `nil`. For example:
                ///
                ///     enum PaperSize: String {
                ///         case A4, A5, Letter, Legal
                ///     }
                ///
                ///     print(PaperSize(rawValue: "Legal"))
                ///     // Prints
"Optional("PaperSize.Legal")"
                ///
                ///     print(PaperSize(rawValue: "Tabloid"))
                ///     // Prints "nil"
                ///
                /// - Parameter rawValue: The raw value to use for
the new instance.
                public init           CFString

                public static let itu_R_709_2
CMFormatDescription Extensions Value ColorPrimaries

                public static let ebu_3213
CMFormatDescription Extensions Value ColorPrimaries

                public static let smpte_C
CMFormatDescription Extensions Value ColorPrimaries

                public static let dci_P3
CMFormatDescription Extensions Value ColorPrimaries

                public static let p3_D65
CMFormatDescription Extensions Value ColorPrimaries

                public static let itu_R_2020
CMFormatDescription Extensions Value ColorPrimaries
```

```
                public static let p22
CMFormatDescription Extensions Value ColorPrimaries


            /// Used for `.colorPrimaries`
            public static func colorPrimaries _

CMFormatDescription Extensions Value ColorPrimaries
CMFormatDescription Extensions Value

            public struct TransferFunction   @unchecked
Sendable

                /// The raw type that can be used to represent all values of
the conforming
                /// type.
                ///
                /// Every distinct value of the conforming type has a
corresponding unique
                /// value of the `RawValue` type, but there may be values
of the `RawValue`
                /// type that don't have a corresponding value of the
conforming type.
                public typealias RawValue   CFString

                /// The corresponding value of the raw type.
                ///
                /// A new instance initialized with `rawValue` will be
equivalent to this
                /// instance. For example:
                ///
                ///     enum PaperSize: String {
                ///         case A4, A5, Letter, Legal
                ///     }
                ///
                ///     let selectedSize = PaperSize.Letter
                ///     print(selectedSize.rawValue)
                ///     // Prints "Letter"
                ///
                ///     print(selectedSize ==
PaperSize(rawValue: selectedSize.rawValue)!)
                ///     // Prints "true"
                public var rawValue  CFString

                /// Creates a new instance with the specified raw value.
                ///
                /// If there is no value of the type that corresponds with the
specified raw
                /// value, this initializer returns `nil`. For example:
```

```
                ///
                ///     enum PaperSize: String {
                ///         case A4, A5, Letter, Legal
                ///     }
                ///
                ///     print(PaperSize(rawValue: "Legal"))
                ///     // Prints
"Optional("PaperSize.Legal")"
                ///
                ///     print(PaperSize(rawValue: "Tabloid"))
                ///     // Prints "nil"
                ///
                /// - Parameter rawValue:  The raw value to use for
the new instance.
                public init           CFString

                public static let itu_R_709_2
CMFormatDescription Extensions Value TransferFunction

                public static let smpte_240M_1995
CMFormatDescription Extensions Value TransferFunction

                public static let useGamma
CMFormatDescription Extensions Value TransferFunction

                ///  Note: semantically equivalent to `itu_R_709_2`, which
is preferred.
                public static let itu_R_2020
CMFormatDescription Extensions Value TransferFunction

                public static let smpte_ST_428_1
CMFormatDescription Extensions Value TransferFunction

                public static let smpte_ST_2084_PQ
CMFormatDescription Extensions Value TransferFunction

                public static let itu_R_2100_HLG
CMFormatDescription Extensions Value TransferFunction

                public static let linear
CMFormatDescription Extensions Value TransferFunction

                public static let sRGB
CMFormatDescription Extensions Value TransferFunction


            ///  Used for `.transferFunction` or
`alternativeTransferCharacteristics`
            public static func transferFunction _
```

```
CMFormatDescription Extensions Value TransferFunction
CMFormatDescription Extensions Value

            public struct YCbCrMatrix   @unchecked Sendable

                /// The raw type that can be used to represent all values of
the conforming
                /// type.
                ///
                /// Every distinct value of the conforming type has a
corresponding unique
                /// value of the `RawValue` type, but there may be values
of the `RawValue`
                /// type that don't have a corresponding value of the
conforming type.
                public typealias RawValue   CFString

                /// The corresponding value of the raw type.
                ///
                /// A new instance initialized with `rawValue` will be
equivalent to this
                /// instance. For example:
                ///
                ///     enum PaperSize: String {
                ///         case A4, A5, Letter, Legal
                ///     }
                ///
                ///     let selectedSize = PaperSize.Letter
                ///     print(selectedSize.rawValue)
                ///     // Prints "Letter"
                ///
                ///     print(selectedSize ==
PaperSize(rawValue: selectedSize.rawValue)!)
                ///     // Prints "true"
                public var rawValue  CFString

                /// Creates a new instance with the specified raw value.
                ///
                /// If there is no value of the type that corresponds with the
specified raw
                /// value, this initializer returns `nil`. For example:
                ///
                ///     enum PaperSize: String {
                ///         case A4, A5, Letter, Legal
                ///     }
                ///
                ///     print(PaperSize(rawValue: "Legal"))
                ///     // Prints
"Optional("PaperSize.Legal")"
                ///
```

```
                    ///     print(PaperSize(rawValue: "Tabloid"))
                    ///     // Prints "nil"
                    ///
                    /// - Parameter rawValue: The raw value to use for
the new instance.
                    public init           CFString

                    public static let itu_R_709_2
CMFormatDescription Extensions Value YCbCrMatrix

                    public static let itu_R_601_4
CMFormatDescription Extensions Value YCbCrMatrix

                    public static let smpted_240M_1995
CMFormatDescription Extensions Value YCbCrMatrix

                    public static let itu_R_2020
CMFormatDescription Extensions Value YCbCrMatrix


                /// Used for `.yCbCrMatrix`
                public static func yCbCrMatrix _
CMFormatDescription Extensions Value YCbCrMatrix
CMFormatDescription Extensions Value

                public struct ChromaLocation   @unchecked Sendable


                    /// The raw type that can be used to represent all values of
the conforming
                    /// type.
                    ///
                    /// Every distinct value of the conforming type has a
corresponding unique
                    /// value of the `RawValue` type, but there may be values
of the `RawValue`
                    /// type that don't have a corresponding value of the
conforming type.
                    public typealias RawValue   CFString

                    /// The corresponding value of the raw type.
                    ///
                    /// A new instance initialized with `rawValue` will be
equivalent to this
                    /// instance. For example:
                    ///
                    ///     enum PaperSize: String {
                    ///         case A4, A5, Letter, Legal
                    ///     }
                    ///
                    ///     let selectedSize = PaperSize.Letter
                    ///     print(selectedSize.rawValue)
                    ///     // Prints "Letter"
                    ///
                    ///     print(selectedSize ==
PaperSize(rawValue: selectedSize.rawValue)!)
                    ///     // Prints "true"
                    public var rawValue  CFString

                    /// Creates a new instance with the specified raw value.
                    ///
                    /// If there is no value of the type that corresponds with the
specified raw
                    /// value, this initializer returns `nil`. For example:
                    ///
                    ///     enum PaperSize: String {
                    ///         case A4, A5, Letter, Legal
                    ///     }
                    ///
                    ///     print(PaperSize(rawValue: "Legal"))
                    ///     // Prints
"Optional("PaperSize.Legal")"
                    ///
                    ///     print(PaperSize(rawValue: "Tabloid"))
                    ///     // Prints "nil"
                    ///
                    /// - Parameter rawValue: The raw value to use for
the new instance.
                    public init           CFString

                    /// Chroma sample is horizontally co-sited with the left
column of luma
                    /// samples, but centered vertically
                    public static let left
CMFormatDescription Extensions Value ChromaLocation

                    /// Chroma sample is fully centered
                    public static let center
CMFormatDescription Extensions Value ChromaLocation

                    /// Chroma sample is co-sited with the top-left luma sample
                    public static let topLeft
CMFormatDescription Extensions Value ChromaLocation

                    /// Chroma sample is horizontally centered, but co-sited with
the top
                    /// row of luma samples
                    public static let top
CMFormatDescription Extensions Value ChromaLocation
```

```
                    ///  Chroma sample is co-sited with the bottom-left luma
sample
                    public static let bottomLeft
CMFormatDescription Extensions Value ChromaLocation

                    ///  Chroma sample is horizontally centered, but co-sited with
the bottom
                    ///  row of luma samples
                    public static let bottom
CMFormatDescription Extensions Value ChromaLocation

                    ///  Cr and Cb samples are alternately co-sited with the left
luma
                    ///  samples of the same field
                    public static let dv420
CMFormatDescription Extensions Value ChromaLocation


                ///  Used for `.chromaLocationTopField` and
`chromaLocationBottomField`
                public static func chromaLocation _

CMFormatDescription Extensions Value ChromaLocation
CMFormatDescription Extensions Value

                public struct MPEG2VideoProfile   @unchecked
Sendable

                    ///  The corresponding value of the raw type.
                    ///
                    ///  A new instance initialized with `rawValue` will be
equivalent to this
                    ///  instance. For example:
                    ///
                    ///      enum PaperSize: String {
                    ///          case A4, A5, Letter, Legal
                    ///      }
                    ///
                    ///      let selectedSize = PaperSize.Letter
                    ///      print(selectedSize.rawValue)
                    ///      // Prints "Letter"
                    ///
                    ///      print(selectedSize ==
PaperSize(rawValue: selectedSize.rawValue)!)
                    ///      // Prints "true"
                    public var rawValue  FourCharCode

                    ///  Creates a new instance with the specified raw value.
                    ///
                    ///  If there is no value of the type that corresponds with the
```

```
specified raw
                    ///  value, this initializer returns `nil`. For example:
                    ///
                    ///      enum PaperSize: String {
                    ///          case A4, A5, Letter, Legal
                    ///      }
                    ///
                    ///      print(PaperSize(rawValue: "Legal"))
                    ///      // Prints
"Optional("PaperSize.Legal")"
                    ///
                    ///      print(PaperSize(rawValue: "Tabloid"))
                    ///      // Prints "nil"
                    ///
                    /// - Parameter rawValue:  The raw value to use for
the new instance.
                    @available macOS 10.15  iOS 13.0  tvOS 13.0
watchOS 6.0  visionOS 1.0
                    public init           FourCharCode

                    @available macOS 10.15  iOS 13.0  tvOS 13.0
watchOS 6.0  visionOS 1.0
                    public init           Int32

                    public static let hdv_720p30
CMFormatDescription Extensions Value MPEG2VideoProfile

                    public static let hdv_1080i60
CMFormatDescription Extensions Value MPEG2VideoProfile

                    public static let hdv_1080i50
CMFormatDescription Extensions Value MPEG2VideoProfile

                    public static let hdv_720p24
CMFormatDescription Extensions Value MPEG2VideoProfile

                    public static let hdv_720p25
CMFormatDescription Extensions Value MPEG2VideoProfile

                    public static let hdv_1080p24
CMFormatDescription Extensions Value MPEG2VideoProfile

                    public static let hdv_1080p25
CMFormatDescription Extensions Value MPEG2VideoProfile

                    public static let hdv_1080p30
CMFormatDescription Extensions Value MPEG2VideoProfile

                    public static let hdv_720p60
CMFormatDescription Extensions Value MPEG2VideoProfile
```

```
                public static let hdv_720p50
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD_1080i60_VBR35
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD_1080i50_VBR35
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD_1080p24_VBR35
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD_1080p25_VBR35
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD_1080p30_VBR35
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_EX_720p24_VBR35
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_EX_720p25_VBR35
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_EX_720p30_VBR35
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_EX_720p50_VBR35
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_EX_720p60_VBR35
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_EX_1080i60_VBR35
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_EX_1080i50_VBR35
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_EX_1080p24_VBR35
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_EX_1080p25_VBR35
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_EX_1080p30_VBR35
CMFormatDescription Extensions Value MPEG2VideoProfile
```

```
                public static let xdcam_HD422_720p50_CBR50
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD422_720p60_CBR50
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD422_1080i60_CBR50
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD422_1080i50_CBR50
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD422_1080p24_CBR50
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD422_1080p25_CBR50
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD422_1080p30_CBR50
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD_540p
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD422_540p
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD422_720p24_CBR50
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD422_720p25_CBR50
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xdcam_HD422_720p30_CBR50
CMFormatDescription Extensions Value MPEG2VideoProfile

                public static let xf
CMFormatDescription Extensions Value MPEG2VideoProfile

                /// The raw type that can be used to represent all values of
the conforming
                /// type.
                ///
                /// Every distinct value of the conforming type has a
corresponding unique
                /// value of the `RawValue` type, but there may be values
of the `RawValue`
                /// type that don't have a corresponding value of the
conforming type.
                @available iOS 13.0  tvOS 13.0  watchOS 6.0
```

```
visionOS 1.0  macOS 10.15
                public typealias RawValue   UInt32


            /// Used for `.conformsToMPEG2VideoProfile`
            public static func mpeg2VideoProfile _

CMFormatDescription Extensions Value MPEG2VideoProfile
CMFormatDescription Extensions Value

            public struct Vendor   @unchecked Sendable

                /// The corresponding value of the raw type.
                ///
                /// A new instance initialized with `rawValue` will be
equivalent to this
                /// instance. For example:
                ///
                ///     enum PaperSize: String {
                ///         case A4, A5, Letter, Legal
                ///     }
                ///
                ///     let selectedSize = PaperSize.Letter
                ///     print(selectedSize.rawValue)
                ///     // Prints "Letter"
                ///
                ///     print(selectedSize ==
PaperSize(rawValue: selectedSize.rawValue)!)
                ///     // Prints "true"
                public var rawValue  CFString

                /// Creates a new instance with the specified raw value.
                ///
                /// If there is no value of the type that corresponds with the
specified raw
                /// value, this initializer returns `nil`. For example:
                ///
                ///     enum PaperSize: String {
                ///         case A4, A5, Letter, Legal
                ///     }
                ///
                ///     print(PaperSize(rawValue: "Legal"))
                ///     // Prints
"Optional("PaperSize.Legal")"
                ///
                ///     print(PaperSize(rawValue: "Tabloid"))
                ///     // Prints "nil"
                ///
                /// - Parameter rawValue: The raw value to use for
the new instance.
```

```
                @available macOS 10.15  iOS 13.0  tvOS 13.0
watchOS 6.0  visionOS 1.0
                public init           CFString

                public static let apple
CMFormatDescription Extensions Value Vendor

                ///  The raw type that can be used to represent all values of
the conforming
                ///  type.
                ///
                ///  Every distinct value of the conforming type has a
corresponding unique
                ///  value of the `RawValue` type, but there may be values
of the `RawValue`
                ///  type that don't have a corresponding value of the
conforming type.
                @available iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0  macOS 10.15
                public typealias RawValue   CFString


            ///  Used for `.vendor`
            public static func vendor _
CMFormatDescription Extensions Value Vendor
CMFormatDescription Extensions Value

            ///  Used for `.vendor`
            public static func vendor _         String
CMFormatDescription Extensions Value

            public struct AlphaChannelMode   @unchecked
Sendable

                ///  The raw type that can be used to represent all values of
the conforming
                ///  type.
                ///
                ///  Every distinct value of the conforming type has a
corresponding unique
                ///  value of the `RawValue` type, but there may be values
of the `RawValue`
                ///  type that don't have a corresponding value of the
conforming type.
                public typealias RawValue   CFString

                ///  The corresponding value of the raw type.
                ///
                ///  A new instance initialized with `rawValue` will be
equivalent to this
```

```
                    /// instance. For example:
                    ///
                    ///     enum PaperSize: String {
                    ///         case A4, A5, Letter, Legal
                    ///     }
                    ///
                    ///     let selectedSize = PaperSize.Letter
                    ///     print(selectedSize.rawValue)
                    ///     // Prints "Letter"
                    ///
                    ///     print(selectedSize ==
PaperSize(rawValue: selectedSize.rawValue)!)
                    ///     // Prints "true"
                    public var rawValue  CFString

                    /// Creates a new instance with the specified raw value.
                    ///
                    /// If there is no value of the type that corresponds with the
specified raw
                    /// value, this initializer returns `nil`. For example:
                    ///
                    ///     enum PaperSize: String {
                    ///         case A4, A5, Letter, Legal
                    ///     }
                    ///
                    ///     print(PaperSize(rawValue: "Legal"))
                    ///     // Prints
"Optional("PaperSize.Legal")"
                    ///
                    ///     print(PaperSize(rawValue: "Tabloid"))
                    ///     // Prints "nil"
                    ///
                    /// - Parameter rawValue: The raw value to use for
the new instance.
                    public init           CFString

                    public static let straightAlpha
CMFormatDescription Extensions Value AlphaChannelMode

                    public static let premultipliedAlpha
CMFormatDescription Extensions Value AlphaChannelMode


                /// Used for `.alphaChannelMode`
                public static func alphaChannelMode _

CMFormatDescription Extensions Value AlphaChannelMode
CMFormatDescription Extensions Value

                /// Used for `.backgroundColor` and
```

```
`qtTextDefaultStyle`
            public static func qtTextColor      CGFloat
       CGFloat        CGFloat         CGFloat
CMFormatDescription Extensions Value

            /// Used for `.backgroundColor` and
`mobile3GPPTextDefaultStyle`
            public static func mobile3GPPTextColor
CGFloat        CGFloat        CGFloat        CGFloat
CMFormatDescription Extensions Value

            /// Used for `.fontTable`
            public static func fontTable _             Int
String      CMFormatDescription Extensions Value

            /// Used for `qtTextDefaultStyle` and
`mobile3GPPTextDefaultStyle`
            public struct FontFace   OptionSet  Sendable

                /// The raw type that can be used to represent all values of
the conforming
                /// type.
                ///
                /// Every distinct value of the conforming type has a
corresponding unique
                /// value of the `RawValue` type, but there may be values
of the `RawValue`
                /// type that don't have a corresponding value of the
conforming type.
                public typealias RawValue   UInt8

                /// The corresponding value of the raw type.
                ///
                /// A new instance initialized with `rawValue` will be
equivalent to this
                /// instance. For example:
                ///
                ///     enum PaperSize: String {
                ///         case A4, A5, Letter, Legal
                ///     }
                ///
                ///     let selectedSize = PaperSize.Letter
                ///     print(selectedSize.rawValue)
                ///     // Prints "Letter"
                ///
                ///     print(selectedSize ==
PaperSize(rawValue: selectedSize.rawValue)!)
                ///     // Prints "true"
                public var rawValue  UInt8
```

```
                    ///  Creates a new option set from the given raw value.
                    ///
                    ///  This initializer always succeeds, even if the value passed
as `rawValue`
                    ///  exceeds the static properties declared as part of the
option set. This
                    ///  example creates an instance of `ShippingOptions`
with a raw value beyond
                    ///  the highest element, with a bit mask that effectively
contains all the
                    ///  declared static members.
                    ///
                    ///      let extraOptions =
ShippingOptions(rawValue: 255)
                    ///
print(extraOptions.isStrictSuperset(of: .all))
                    ///      // Prints "true"
                    ///
                    /// - Parameter rawValue:  The raw value of the option
set to create. Each bit
                    ///     of `rawValue` potentially represents an element of
the option set,
                    ///     though raw values may include bits that are not
defined as distinct
                    ///     values of the `OptionSet` type.
                    public init           UInt8

                    public static let bold
CMFormatDescription Extensions Value FontFace

                    public static let italic
CMFormatDescription Extensions Value FontFace

                    public static let underline
CMFormatDescription Extensions Value FontFace

                    public static let all
CMFormatDescription Extensions Value FontFace

                    ///  The type of the elements of an array literal.
                    @available iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0  macOS 10.15
                    public typealias ArrayLiteralElement
CMFormatDescription Extensions Value FontFace

                    ///  The element type of the option set.
                    ///
                    ///  To inherit all the default implementations from the
`OptionSet` protocol,
                    ///  the `Element` type must be `Self`, the default.
```

```
                @available iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0  macOS 10.15
                public typealias Element
CMFormatDescription Extensions Value FontFace


            /// Used for `.defaultStyle`
            public static func qtTextDefaultStyle
Int             Int             Int                  Int
CMFormatDescription Extensions Value FontFace             Int
                 CMFormatDescription Extensions Value
                 String
CMFormatDescription Extensions Value

            /// Used for `.defaultStyle`
            public static func
mobile3GPPTextDefaultStyle            Int            Int
             Int
CMFormatDescription Extensions Value FontFace             Int
                 CMFormatDescription Extensions Value
CMFormatDescription Extensions Value

            /// Used for `.defaultTextBox`
            public static func textRect      Int        Int
        Int         Int
CMFormatDescription Extensions Value

            /// Display mode flags for text media
            public struct TextDisplayFlags   Sendable

                /// The raw type that can be used to represent all values of
the conforming
                /// type.
                ///
                /// Every distinct value of the conforming type has a
corresponding unique
                /// value of the `RawValue` type, but there may be values
of the `RawValue`
                /// type that don't have a corresponding value of the
conforming type.
                public typealias RawValue   CMTextDisplayFlags

                /// The corresponding value of the raw type.
                ///
                /// A new instance initialized with `rawValue` will be
equivalent to this
                /// instance. For example:
                ///
                ///     enum PaperSize: String {
                ///         case A4, A5, Letter, Legal
                ///     }
                ///
                ///     let selectedSize = PaperSize.Letter
                ///     print(selectedSize.rawValue)
                ///     // Prints "Letter"
                ///
                ///     print(selectedSize ==
PaperSize(rawValue: selectedSize.rawValue)!)
                ///     // Prints "true"
                public var rawValue  CMTextDisplayFlags

                /// Creates a new instance with the specified raw value.
                ///
                /// If there is no value of the type that corresponds with the
specified raw
                /// value, this initializer returns `nil`. For example:
                ///
                ///     enum PaperSize: String {
                ///         case A4, A5, Letter, Legal
                ///     }
                ///
                ///     print(PaperSize(rawValue: "Legal"))
                ///     // Prints
"Optional("PaperSize.Legal")"
                ///
                ///     print(PaperSize(rawValue: "Tabloid"))
                ///     // Prints "nil"
                ///
                /// - Parameter rawValue: The raw value to use for
the new instance.
                public init           CMTextDisplayFlags

                /// Text scrolls into the display region.
                public static let scrollIn
CMFormatDescription Extensions Value TextDisplayFlags

                /// Text scrolls out of the display region.
                public static let scrollOut
CMFormatDescription Extensions Value TextDisplayFlags

                /// The scrolling direction is set by a two-bit field, obtained
from
                /// displayFlags using `scrollDirectionMask`.
                public static let scrollDirectionMask
CMFormatDescription Extensions Value TextDisplayFlags

                /// Text is vertically scrolled up ("credits style"), entering from
the
                /// bottom and leaving towards the top.
                public static let scrollDirection_bottomToTop
```

```
CMFormatDescription Extensions Value TextDisplayFlags

                        ///  Text is horizontally scrolled ("marquee style"), entering
from the
                        ///  right and leaving towards the left.
                        public static let scrollDirection_rightToLeft
CMFormatDescription Extensions Value TextDisplayFlags

                        ///  Text is vertically scrolled down, entering from the top and
leaving
                        ///  towards the bottom.
                        public static let scrollDirection_topToBottom
CMFormatDescription Extensions Value TextDisplayFlags

                        ///  Text is horizontally scrolled, entering from the left and
leaving
                        ///  towards the right.
                        public static let scrollDirection_leftToRight
CMFormatDescription Extensions Value TextDisplayFlags

                        ///  Enables the Continuous Karaoke mode where the range
of karaoke
                        ///  highlighting extends to include additional ranges rather
than the
                        ///  highlighting moves onto the next range.
                        public static let continuousKaraoke
CMFormatDescription Extensions Value TextDisplayFlags

                        ///  Specifies the text to be rendered vertically.
                        public static let writeTextVertically
CMFormatDescription Extensions Value TextDisplayFlags

                        ///  The subtitle display bounds are to be filled with the color
                        ///  specified by `.backgroundColor`.
                        public static let fillTextRegion
CMFormatDescription Extensions Value TextDisplayFlags

                        ///  Specifies that the subtitle display bounds should be used
to
                        ///  determine if the subtitles should be placed near the top or
the
                        ///  bottom of the video. Otherwise, subtitles should be
placed at the
                        ///  bottom of the video.
                        public static let obeySubtitleFormatting
CMFormatDescription Extensions Value TextDisplayFlags

                        ///  There are forced subtitles present, e.g., a subtitle which
only
                        ///  displays during foreign language sections of the video.
Check
```

```
                /// individual samples to determine what type of subtitle is
contained.
                public static let forcedSubtitlesPresent
CMFormatDescription Extensions Value TextDisplayFlags

                /// Treat all subtitle samples as if they contain forced
subtitles.
                public static let allSubtitlesForced
CMFormatDescription Extensions Value TextDisplayFlags

                /// The scrollDirection part of this `TextDisplayFlags`
                public var scrollDirection
CMFormatDescription Extensions Value TextDisplayFlags    get


            /// Used for `.displayFlags`
            public static func textDisplayFlags _

Set CMFormatDescription Extensions Value TextDisplayFlags
CMFormatDescription Extensions Value

            /// Justification modes for text media. Used when specifying either
            /// horizontal or vertical justification.
            public struct TextJustification    Sendable

                /// The raw type that can be used to represent all values of
the conforming
                /// type.
                ///
                /// Every distinct value of the conforming type has a
corresponding unique
                /// value of the `RawValue` type, but there may be values
of the `RawValue`
                /// type that don't have a corresponding value of the
conforming type.
                public typealias RawValue    Int8

                /// The corresponding value of the raw type.
                ///
                /// A new instance initialized with `rawValue` will be
equivalent to this
                /// instance. For example:
                ///
                ///     enum PaperSize: String {
                ///         case A4, A5, Letter, Legal
                ///     }
                ///
                ///     let selectedSize = PaperSize.Letter
                ///     print(selectedSize.rawValue)
                ///     // Prints "Letter"
                ///
                ///     print(selectedSize ==
PaperSize(rawValue: selectedSize.rawValue)!)
                ///     // Prints "true"
                public var rawValue  Int8

                /// Creates a new instance with the specified raw value.
                ///
                /// If there is no value of the type that corresponds with the
specified raw
                /// value, this initializer returns `nil`. For example:
                ///
                ///     enum PaperSize: String {
                ///         case A4, A5, Letter, Legal
                ///     }
                ///
                ///     print(PaperSize(rawValue: "Legal"))
                ///     // Prints
"Optional("PaperSize.Legal")"
                ///
                ///     print(PaperSize(rawValue: "Tabloid"))
                ///     // Prints "nil"
                ///
                /// - Parameter rawValue: The raw value to use for
the new instance.
                public init           Int8

                /// Left justification
                public static let left
CMFormatDescription Extensions Value TextJustification

                /// Top justification
                public static let top
CMFormatDescription Extensions Value TextJustification

                /// Center justification
                public static let centered
CMFormatDescription Extensions Value TextJustification

                /// Bottom justification
                public static let bottom
CMFormatDescription Extensions Value TextJustification

                /// Right justification
                public static let right
CMFormatDescription Extensions Value TextJustification


            /// Used for `.horizontalJustification`,
`.verticalJustification`
```

```
            /// (`.mobile3GPP`) or `.textJustification` (`.qt`)
            public static func textJustification _
CMFormatDescription Extensions Value TextJustification
CMFormatDescription Extensions Value

            /// Used for `.sourceReferenceName`
            public static func sourceReferenceName
String            Int     CMFormatDescription Extensions Value


        /// Creates an empty `Extensions` structure.
        public init

        /// Creates an `Extensions` structure with existing values.
        public init        CFString   CFPropertyList

        /// Accesses values using a predefined `Key`
        public subscript
CMFormatDescription Extensions Key
CMFormatDescription Extensions Value

        public subscript      CFString     CFPropertyList



@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMFormatDescription

    /// CMFormatDescription Errors
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct Error   Sendable

        /// Invalid parameter.
        public static let invalidParameter  NSError

        /// Thrown when an allocation fails.
        public static let allocationFailed  NSError

        /// Thrown when the `CMFormatDescription` does not carry such
a value.
        public static let valueNotAvailable  NSError



@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
```

```
extension CMFormatDescription

    /// The type of media described by a `CMFormatDescription`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct MediaType   Sendable

        /// The corresponding value of the raw type.
        ///
        /// A new instance initialized with `rawValue` will be equivalent to this
        /// instance. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter, Legal
        ///     }
        ///
        ///     let selectedSize = PaperSize.Letter
        ///     print(selectedSize.rawValue)
        ///     // Prints "Letter"
        ///
        ///     print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue)!)
        ///     // Prints "true"
        public var rawValue  CMMediaType

        /// Creates a new instance with the specified raw value.
        ///
        /// If there is no value of the type that corresponds with the specified
raw
        /// value, this initializer returns `nil`. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter, Legal
        ///     }
        ///
        ///     print(PaperSize(rawValue: "Legal"))
        ///     // Prints "Optional("PaperSize.Legal")"
        ///
        ///     print(PaperSize(rawValue: "Tabloid"))
        ///     // Prints "nil"
        ///
        /// - Parameter rawValue: The raw value to use for the new
instance.
        @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS
6.0  visionOS 1.0
        public init           CMMediaType

        /// Video media
        public static let video  CMFormatDescription MediaType
        /// Audio media
        public static let audio  CMFormatDescription MediaType

        /// Muxed media
        public static let muxed  CMFormatDescription MediaType

        /// Text media
        public static let text  CMFormatDescription MediaType

        /// Closed-caption media
        public static let closedCaption
CMFormatDescription MediaType

        /// Subtitle media
        public static let subtitle
CMFormatDescription MediaType

        /// TimeCode media
        public static let timeCode
CMFormatDescription MediaType

        /// Metadata media
        public static let metadata
CMFormatDescription MediaType

        /// Tagged buffer group media
        public static let taggedBufferGroup
CMFormatDescription MediaType

        /// The raw type that can be used to represent all values of the
conforming
        /// type.
        ///
        /// Every distinct value of the conforming type has a corresponding
unique
        /// value of the `RawValue` type, but there may be values of the
`RawValue`
        /// type that don't have a corresponding value of the conforming type.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias RawValue   UInt32



@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMFormatDescription

    /// MediaSubType
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
```

```
visionOS 1.0
    public struct MediaSubType   Sendable

        /// The corresponding value of the raw type.
        ///
        /// A new instance initialized with `rawValue` will be equivalent to this
        /// instance. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter, Legal
        ///     }
        ///
        ///     let selectedSize = PaperSize.Letter
        ///     print(selectedSize.rawValue)
        ///     // Prints "Letter"
        ///
        ///     print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue)!)
        ///     // Prints "true"
        public var rawValue  FourCharCode

        /// Creates a new instance with the specified raw value.
        ///
        /// If there is no value of the type that corresponds with the specified
raw
        /// value, this initializer returns `nil`. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter, Legal
        ///     }
        ///
        ///     print(PaperSize(rawValue: "Legal"))
        ///     // Prints "Optional("PaperSize.Legal")"
        ///
        ///     print(PaperSize(rawValue: "Tabloid"))
        ///     // Prints "nil"
        ///
        /// - Parameter rawValue: The raw value to use for the new
instance.
        @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS
6.0  visionOS 1.0
        public init           FourCharCode

        /// Certain codec types are also audio formats.
        public static let linearPCM
CMFormatDescription MediaSubType

        public static let ac3
CMFormatDescription MediaSubType
```

```
        public static let iec60958AC3
CMFormatDescription MediaSubType

        public static let appleIMA4
CMFormatDescription MediaSubType

        public static let mpeg4AAC
CMFormatDescription MediaSubType

        public static let mpeg4CELP
CMFormatDescription MediaSubType

        public static let mpeg4HVXC
CMFormatDescription MediaSubType

        public static let mpeg4TwinVQ
CMFormatDescription MediaSubType

        public static let mace3
CMFormatDescription MediaSubType

        public static let mace6
CMFormatDescription MediaSubType

        public static let uLaw
CMFormatDescription MediaSubType

        public static let aLaw
CMFormatDescription MediaSubType

        public static let qDesign
CMFormatDescription MediaSubType

        public static let qDesign2
CMFormatDescription MediaSubType

        public static let qualcomm
CMFormatDescription MediaSubType

        public static let mpegLayer1
CMFormatDescription MediaSubType

        public static let mpegLayer2
CMFormatDescription MediaSubType

        public static let mpegLayer3
CMFormatDescription MediaSubType

        public static let timeCode
```

```
CMFormatDescription MediaSubType

        public static let midiStream
CMFormatDescription MediaSubType

        public static let parameterValueStream
CMFormatDescription MediaSubType

        public static let appleLossless
CMFormatDescription MediaSubType

        public static let mpeg4AAC_HE
CMFormatDescription MediaSubType

        public static let mpeg4AAC_LD
CMFormatDescription MediaSubType

        public static let mpeg4AAC_ELD
CMFormatDescription MediaSubType

        public static let mpeg4AAC_ELD_SBR
CMFormatDescription MediaSubType

        public static let mpeg4AAC_ELD_V2
CMFormatDescription MediaSubType

        public static let mpeg4AAC_HE_V2
CMFormatDescription MediaSubType

        public static let mpeg4AAC_Spatial
CMFormatDescription MediaSubType

        public static let mpegD_USAC
CMFormatDescription MediaSubType

        public static let amr
CMFormatDescription MediaSubType

        public static let amr_WB
CMFormatDescription MediaSubType

        public static let audible
CMFormatDescription MediaSubType

        public static let iLBC
CMFormatDescription MediaSubType

        public static let dviIntelIMA
CMFormatDescription MediaSubType
```

```
        public static let microsoftGSM
CMFormatDescription MediaSubType

        public static let aes3
CMFormatDescription MediaSubType

        public static let enhancedAC3
CMFormatDescription MediaSubType

        public static let flac
CMFormatDescription MediaSubType

        public static let opus
CMFormatDescription MediaSubType

        /// iTMS protected low-complexity AAC.
        public static let aacLCProtected
CMFormatDescription MediaSubType

        /// Audible's protected AAC.
        public static let aacAudibleProtected
CMFormatDescription MediaSubType

        /// 32-bit ARGB
        public static let pixelFormat_32ARGB
CMFormatDescription MediaSubType

        /// 32-bit BGRA
        public static let pixelFormat_32BGRA
CMFormatDescription MediaSubType

        /// 24-bit RGB
        public static let pixelFormat_24RGB
CMFormatDescription MediaSubType

        /// 16-bit big-endian 5-5-5
        public static let pixelFormat_16BE555
CMFormatDescription MediaSubType

        /// 16-bit big-endian 5-6-5
        public static let pixelFormat_16BE565
CMFormatDescription MediaSubType

        /// 16-bit little-endian 5-5-5
        public static let pixelFormat_16LE555
CMFormatDescription MediaSubType

        /// 16-bit little-endian 5-6-5
        public static let pixelFormat_16LE565
CMFormatDescription MediaSubType

        ///  16-bit little-endian 5-5-5-1
        public static let pixelFormat_16LE5551
CMFormatDescription MediaSubType

        ///  Component Y'CbCr 8-bit 4:2:2 ordered Cb Y'0 Cr Y'1
        public static let pixelFormat_422YpCbCr8
CMFormatDescription MediaSubType

        ///  Component Y'CbCr 8-bit 4:2:2 ordered Y'0 Cb Y'1 Cr
        public static let pixelFormat_422YpCbCr8_yuvs
CMFormatDescription MediaSubType

        ///  Component Y'CbCr 8-bit 4:4:4
        public static let pixelFormat_444YpCbCr8
CMFormatDescription MediaSubType

        ///  Component Y'CbCrA 8-bit 4:4:4:4
        public static let pixelFormat_4444YpCbCrA8
CMFormatDescription MediaSubType

        ///  Component Y'CbCr 10,12,14,16-bit 4:2:2
        public static let pixelFormat_422YpCbCr16
CMFormatDescription MediaSubType

        ///  Component Y'CbCr 10-bit 4:2:2
        public static let pixelFormat_422YpCbCr10
CMFormatDescription MediaSubType

        ///  Component Y'CbCr 10-bit 4:4:4
        public static let pixelFormat_444YpCbCr10
CMFormatDescription MediaSubType

        ///  8 bit indexed gray, white is zero
        public static let
pixelFormat_8IndexedGray_WhiteIsZero
CMFormatDescription MediaSubType

        ///  Apple Animation format
        public static let animation
CMFormatDescription MediaSubType

        ///  Cinepak format
        public static let cinepak
CMFormatDescription MediaSubType

        ///  Joint Photographic Experts Group (JPEG) format
        public static let jpeg
CMFormatDescription MediaSubType

        /// JPEG format with Open-DML extensions
        public static let jpeg_OpenDML
CMFormatDescription MediaSubType

        /// Sorenson video format
        public static let sorensonVideo
CMFormatDescription MediaSubType

        /// Sorenson 3 video format
        public static let sorensonVideo3
CMFormatDescription MediaSubType

        /// ITU-T H.263 format
        public static let h263
CMFormatDescription MediaSubType

        /// ITU-T H.264 format (AKA ISO/IEC 14496-10 - MPEG-4 Part 10,
Advanced Video Coding format)
        public static let h264
CMFormatDescription MediaSubType

        /// ITU-T HEVC format
        public static let hevc
CMFormatDescription MediaSubType

        /// HEVC format with alpha support defined in Annex-F.
        ///
        /// IMPORTANT NOTE: this constant is used to select the appropriate
        /// encoder, but is NOT used on the encoded content, which is
backwards
        /// compatible and hence uses `'hvc1'` as its codec type.
        public static let hevcWithAlpha
CMFormatDescription MediaSubType

        /// ISO/IEC Moving Picture Experts Group (MPEG) MPEG-4 Part 2
video format
        public static let mpeg4Video
CMFormatDescription MediaSubType

        /// MPEG-2 video format
        public static let mpeg2Video
CMFormatDescription MediaSubType

        /// MPEG-1 video format
        public static let mpeg1Video
CMFormatDescription MediaSubType
```

```
        ///  DV NTSC format
        public static let dvcNTSC
CMFormatDescription MediaSubType

        ///  DV PAL format
        public static let dvcPAL
CMFormatDescription MediaSubType

        ///  Panasonic DVCPro PAL format
        public static let dvcProPAL
CMFormatDescription MediaSubType

        ///  Panasonic DVCPro-50 NTSC format
        public static let dvcPro50NTSC
CMFormatDescription MediaSubType

        ///  Panasonic DVCPro-50 PAL format
        public static let dvcPro50PAL
CMFormatDescription MediaSubType

        ///  Panasonic DVCPro-HD 720p60 format
        public static let dvcPROHD720p60
CMFormatDescription MediaSubType

        ///  Panasonic DVCPro-HD 720p50 format
        public static let dvcPROHD720p50
CMFormatDescription MediaSubType

        ///  Panasonic DVCPro-HD 1080i60 format
        public static let dvcPROHD1080i60
CMFormatDescription MediaSubType

        ///  Panasonic DVCPro-HD 1080i50 format
        public static let dvcPROHD1080i50
CMFormatDescription MediaSubType

        ///  Panasonic DVCPro-HD 1080p30 format
        public static let dvcPROHD1080p30
CMFormatDescription MediaSubType

        ///  Panasonic DVCPro-HD 1080p25 format
        public static let dvcPROHD1080p25
CMFormatDescription MediaSubType

        ///  Apple ProRes 4444 XQ format
        public static let proRes4444XQ
CMFormatDescription MediaSubType

        ///  Apple ProRes 4444 format
        public static let proRes4444
CMFormatDescription MediaSubType

        /// Apple ProRes 422 HQ format
        public static let proRes422HQ
CMFormatDescription MediaSubType

        /// Apple ProRes 422 format
        public static let proRes422
CMFormatDescription MediaSubType

        /// Apple ProRes 422 LT format
        public static let proRes422LT
CMFormatDescription MediaSubType

        /// Apple ProRes 422 Proxy format
        public static let proRes422Proxy
CMFormatDescription MediaSubType

        /// Apple ProRes RAW format
        public static let proResRAW
CMFormatDescription MediaSubType

        /// Apple ProRes RAW HQ format
        public static let proResRAWHQ
CMFormatDescription MediaSubType

        /// MPEG-1 System stream
        public static let mpeg1System
CMFormatDescription MediaSubType

        /// MPEG-2 Transport stream
        public static let mpeg2Transport
CMFormatDescription MediaSubType

        /// MPEG-2 Program stream
        public static let mpeg2Program
CMFormatDescription MediaSubType

        /// DV stream
        public static let dv  CMFormatDescription MediaSubType

        /// iOS Screen capture
        @available macOS 14.0  iOS 17.0  tvOS 17.0  watchOS
10.0  visionOS 1.0
        public static let embeddedDeviceScreenRecording
CMFormatDescription MediaSubType

        /// Closed caption subtypes
        ///  CEA 608-compliant samples
        public static let cea608
CMFormatDescription MediaSubType

        ///  CEA 708-compliant samples
        public static let cea708
CMFormatDescription MediaSubType

        ///  ATSC/52 part-4 compliant samples
        public static let atsc
CMFormatDescription MediaSubType

        ///  Text subtypes
        ///  QuickTime Text media
        public static let qt  CMFormatDescription MediaSubType

        ///  3GPP Text media
        public static let mobile3GPP
CMFormatDescription MediaSubType

        ///  Subtitle subtypes
        public static let webVTT
CMFormatDescription MediaSubType

        ///  TimeCode subtypes
        ///  32-bit timeCode sample.
        public static let timeCode32
CMFormatDescription MediaSubType

        ///  64-bit timeCode sample.
        public static let timeCode64
CMFormatDescription MediaSubType

        ///  32-bit counter-mode sample.
        public static let counter32
CMFormatDescription MediaSubType

        ///  64-bit counter-mode sample.
        public static let counter64
CMFormatDescription MediaSubType

        ///  Metadata subtypes
        ///  SHOUTCast format.
        public static let icy
CMFormatDescription MediaSubType

        ///  ID3 format.
        public static let id3
CMFormatDescription MediaSubType
```

```
        /// Boxed format.
        public static let boxed
CMFormatDescription MediaSubType

        /// EMSG format.
        public static let emsg
CMFormatDescription MediaSubType

        /// TaggedBufferGroup format.
        public static let tbgr
CMFormatDescription MediaSubType

        /// The raw type that can be used to represent all values of the
conforming
        /// type.
        ///
        /// Every distinct value of the conforming type has a corresponding
unique
        /// value of the `RawValue` type, but there may be values of the
`RawValue`
        /// type that don't have a corresponding value of the conforming type.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias RawValue   UInt32



@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMFormatDescription

    /// The `CFTypeID` corresponding to `CMFormatDescription`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static var typeID  CFTypeID   get


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMFormatDescription

    /// Compares two format descriptions for equality, ignoring differences in
    /// specified lists of format description extension keys and sample
    /// description extension keys.
    ///
    /// If any keys are passed,
    ///
`kCMFormatDescriptionExtension_VerbatimSampleDescription` and
    /// `kCMFormatDescriptionExtension_VerbatimISOSampleEntry`
```

```
will also be
    ///  automatically ignored for the purpose of comparison.
    ///
    /// - Parameters:
    ///   - otherFormatDescription:  A format description to compare to.
    ///   - extensionKeysToIgnore:  An array of format description
extension keys.
    ///   - sampleDescriptionExtensionAtomKeysToIgnore:  An array
of sample
    ///      description extension atom keys. See
    ///
`kCMFormatDescriptionExtension_SampleDescriptionExtensionAtoms
`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func equalTo _
CMFormatDescription
 CMFormatDescription Extensions Key
                                             String
Bool

    ///  The media type.
    ///
    ///  For example, returns `.audio` for a description of an audio stream.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var mediaType  CMFormatDescription MediaType
get

    ///  The media subtype of the CMFormatDescription.
    ///
    ///  The media subtype is defined in a media-specific way.
    ///  For audio streams, the media subtype is the `asbd.mFormatID`.
    ///  For video streams, the media subtype is the video codec type.
    ///  For muxed streams, it is the format of the muxed stream.
    ///  For example, `'aac '` is returned for a description of an AAC audio
stream,
    ///  `'avc1'` is returned for a description of an H.264 video stream, and
    ///  `'mp2t'` is returned for a description of an MPEG-2 transport (muxed)
    ///  stream.
    ///
    ///  If a particular type of media stream does not have subtypes, this returns
    ///  `0`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var mediaSubType  CMFormatDescription MediaSubType
  get

    ///  An immutable dictionary containing all the extensions.
    ///
    /// Extensions dictionaries are valid property list objects. This means that
    /// dictionary keys are all `CFStrings`, and the values are all either
    /// `CFNumber`, `CFString`, `CFBoolean`, `CFArray`,
`CFDictionary`, `CFDate`,
    /// or `CFData`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var extensions  CMFormatDescription Extensions
get


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMFormatDescription

    /// Equality is derived from
    /// ```
    /// lhs.equalTo(rhs,
    ///             extensionKeysToIgnore: [],
    ///
sampleDescriptionExtensionAtomKeysToIgnore: [])
    /// ```
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static func          CMFormatDescription
CMFormatDescription     Bool


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMFormatDescription

    /// Copy of the `AudioStreamBasicDescription`.
    ///
    /// See `CoreAudioTypes.h` for the definition of
AudioStreamBasicDescription`.
    ///
    /// This API is specific to audio format descriptions, and will return `nil`
    /// if used with a non-audio format description.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var audioStreamBasicDescription
AudioStreamBasicDescription     get

    /// Get access to the magic cookie.
    ///
    /// The magic cookie is a completely opaque piece of data, written and read
    /// only by the codec itself. A magic cookie is only present for codecs that
    /// require it; this API will return `nil` if one does not exist. This API is
```

```
    /// specific to audio format descriptions, and will return `nil` if called
    /// with a non-audio format description.
    ///
    /// - Parameter body: A closure with an
`UnsafeRawBufferPointer` parameter
    ///     that points to the magic cookie in the audio format description.
    /// - Returns: The return value, if any, of the body closure parameter.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func withMagicCookie R   _
 UnsafeRawBufferPointer   throws    R  rethrows    R

    /// Copy of the magic cookie.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var magicCookie  Data    get

    /// The `AudioChannelLayout`.
    ///
    /// See `CoreAudioTypes.h` for the definition of
`AudioChannelLayout`.
    ///
    /// Audio channel layouts are optional; this API will return `nil` if one does
    /// not exist. This API is specific to audio format descriptions, and will
    /// return `nil` if called with a non-audio format description.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var audioChannelLayout  ManagedAudioChannelLayout
  get

    /// List of `AudioFormatListItem` structs describing the audio formats
    /// contained within the format description.
    ///
    /// This property is analogous to
`kAudioFormatProperty_FormatList` (See
    /// AudioFormat.h) and follows its conventions.
    ///
    /// Namely, formats are returned in order from the most to least 'rich', with
    /// channel count taking the highest precedence followed by sample rate.
    /// This API is specific to audio format descriptions, and will return an
    /// empty array if called with a non-audio format description.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var audioFormatList   AudioFormatListItem    get

    /// Richest `AudioFormatListItem` inside an audio format description.
    ///
    /// This property performs validation on the formats represented by the audio
    /// in the description.
    ///
    /// It finds the first `AudioFormatListItem` for which the current system
has
    /// a valid decoder.
    ///
    /// This API is specific to audio format descriptions, and will return `nil`
    /// if called with a non-audio format description.
    ///
    /// It may also return `nil` if there is no suitable decoder available on the
    /// current system for this audio format.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var richestDecodableFormat  AudioFormatListItem
get

    /// Most compatible `AudioFormatListItem` inside an audio format
description.
    ///
    /// This property returns a pointer to the last `AudioFormatListItem` in
the
    /// `kAudioFormatProperty_FormatList` (see `AudioFormat.h`).
    ///
    /// This API is specific to audio format descriptions, and will return `nil`
    /// if called with a non-audio format description.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var mostCompatibleFormat  AudioFormatListItem
get


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMFormatDescription

    /// Mask bits passed to (and returned from)
    /// `equalTo(otherFormatDescription:equalityMask:)`,
representing various
    /// parts of an audio format description.
    public struct EqualityMask   OptionSet  Sendable

        /// The corresponding value of the raw type.
        ///
        /// A new instance initialized with `rawValue` will be equivalent to this
        /// instance. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter, Legal
        ///     }
        ///
        ///     let selectedSize = PaperSize.Letter
        ///     print(selectedSize.rawValue)
        ///     // Prints "Letter"
        ///
        ///     print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue)!)
        ///     // Prints "true"
        public let rawValue  CMAudioFormatDescriptionMask

        /// Creates a new option set from the given raw value.
        ///
        /// This initializer always succeeds, even if the value passed as
`rawValue`
        /// exceeds the static properties declared as part of the option set. This
        /// example creates an instance of `ShippingOptions` with a raw
value beyond
        /// the highest element, with a bit mask that effectively contains all the
        /// declared static members.
        ///
        ///     let extraOptions = ShippingOptions(rawValue:
255)
        ///     print(extraOptions.isStrictSuperset(of: .all))
        ///     // Prints "true"
        ///
        /// - Parameter rawValue: The raw value of the option set to
create. Each bit
        ///   of `rawValue` potentially represents an element of the option
set,
        ///   though raw values may include bits that are not defined as distinct
        ///   values of the `OptionSet` type.
        public init            CMAudioFormatDescriptionMask

        /// Represents the `AudioStreamBasicDescription`.
        public static let streamBasicDescription
CMFormatDescription EqualityMask

        /// Represents the magic cookie.
        public static let magicCookie
CMFormatDescription EqualityMask

        /// Represents the `AudioChannelLayout`.
        public static let channelLayout
CMFormatDescription EqualityMask

        /// Represents the format description extensions.
        public static let extensions
CMFormatDescription EqualityMask

        /// Represents all the parts of an audio format description.
        public static let all
CMFormatDescription EqualityMask
```

```
        /// The type of the elements of an array literal.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias ArrayLiteralElement
CMFormatDescription EqualityMask

        /// The element type of the option set.
        ///
        /// To inherit all the default implementations from the `OptionSet`
protocol,
        /// the `Element` type must be `Self`, the default.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias Element
CMFormatDescription EqualityMask

        /// The raw type that can be used to represent all values of the
conforming
        /// type.
        ///
        /// Every distinct value of the conforming type has a corresponding
unique
        /// value of the `RawValue` type, but there may be values of the
`RawValue`
        /// type that don't have a corresponding value of the conforming type.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias RawValue
CMAudioFormatDescriptionMask


    /// Evaluates equality for the specified parts of two audio format
    /// descriptions.
    ///
    /// Bits in `equalityMask` specify the caller's interest in the equality of
    /// various parts of the descriptions.
    ///
    /// If there is any sort of error that prevents the comparison from occurring,
    /// `false` will be returned, and all bits in `equalityMask` will be
cleared.
    /// If you pass `.all` in `equalityMask`, and _ for `equalityMask`, this
API
    /// is equivalent to CFEqual(desc1, desc2).
    ///
    /// See CMAudioFormatDescriptionEqual
    ///
    /// - Parameters:
    ///   - otherFormatDescription: The CMAudioFormatDescription to
which the
```

```
    /// comparison is done.
    ///    - equalityMask: Mask specifying which parts of the descriptions to
    /// compare.
    ///
    /// - Returns: `true` if all parts in which the caller is interested are
    /// equal. `false` if any of the parts in which the caller is interested are
    /// not equal. Bits set and returned in `equalityMask` represent the
subset of
    /// those parts that are equal.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func equalTo _
CMAudioFormatDescription
CMFormatDescription EqualityMask            Bool
              CMFormatDescription EqualityMask


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMFormatDescription

    /// Collection of parameter sets in a video format description.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct ParameterSetCollection
RandomAccessCollection

        /// A type representing the sequence's elements.
        public typealias Element   Data

        /// A type that represents a position in the collection.
        ///
        /// Valid indices consist of the position of every element and a
        /// "past the end" position that's not valid for use as a subscript
        /// argument.
        public typealias Index   Int

        /// The position of the first element in a nonempty collection.
        ///
        /// If the collection is empty, `startIndex` is equal to `endIndex`.
        public var startIndex  Int   get

        /// The collection's "past the end" position---that is, the position one
        /// greater than the last valid subscript argument.
        ///
        /// When you need a range that includes the last element of a
collection, use
        /// the half-open range operator (`..<`) with `endIndex`. The
`..<` operator
        /// creates a range that doesn't include the upper bound, so it's always
        ///  safe to use with `endIndex`. For example:
        ///
        ///      let numbers = [10, 20, 30, 40, 50]
        ///      if let index = numbers.firstIndex(of: 30) {
        ///          print(numbers[index ..< numbers.endIndex])
        ///      }
        ///      // Prints "[30, 40, 50]"
        ///
        ///  If the collection is empty, `endIndex` is equal to `startIndex`.
        public var endIndex  Int   get

        ///  Accesses the element at the specified position.
        ///
        ///  The following example accesses an element of an array through its
        ///  subscript to print its value:
        ///
        ///      var streets = ["Adams", "Bryant", "Channing",
"Douglas", "Evarts"]
        ///      print(streets[1])
        ///      // Prints "Bryant"
        ///
        ///  You can subscript a collection with any valid index other than the
        ///  collection's end index. The end index refers to the position one past
        ///  the last element of a collection, so it doesn't correspond with an
        ///  element.
        ///
        /// - Parameter position:  The position of the element to access.
`position`
        ///    must be a valid index of the collection that is not equal to the
        ///    `endIndex` property.
        ///
        /// - Complexity:  O(1)
        public subscript          Int      Data    get

        ///  A type that represents the indices that are valid for subscripting the
        ///  collection, in ascending order.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias Indices
Range CMFormatDescription ParameterSetCollection Index

        ///  A type that provides the collection's iteration interface and
        ///  encapsulates its iteration state.
        ///
        ///  By default, a collection conforms to the `Sequence` protocol by
        ///  supplying `IndexingIterator` as its associated `Iterator`
        ///  type.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
```

```
        public typealias Iterator
IndexingIterator CMFormatDescription ParameterSetCollection

        /// A collection representing a contiguous subrange of this collection's
        /// elements. The subsequence shares indices with the original
collection.
        ///
        /// The default subsequence type for collections that don't define their
own
        /// is `Slice`.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias SubSequence
Slice CMFormatDescription ParameterSetCollection



@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMFormatDescription

    /// Size, in bytes, of the `NALUnitLength` field in an AVC or HEVC video
    /// sample or parameter set sample.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var nalUnitHeaderLength  Int    get

    /// Parameter sets contained in a H.264 or HEVC format description.
    ///
    /// Parameter sets are parsed from NAL unit in the decoder configuration
    /// record contained in a video format description. These NAL units are
    /// typically parameter sets (e.g. VPS, SPS, PPS), but may contain others as
    /// specified by ISO/IEC 14496-15 (e.g. user-data SEI).
    ///
    /// The parameter set NAL units returned will already have any emulation
    /// prevention bytes needed.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var parameterSets
CMFormatDescription ParameterSetCollection   get

    /// Dimensions in encoded pixels.
    ///
    /// This does not take into account pixel aspect ratio or clean aperture tags.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var dimensions  CMVideoDimensions   get

    /// Returns the dimensions, adjusted to take pixel aspect ratio and/or clean
    /// aperture into account.
    ///
    /// Pixel aspect ratio is used to adjust the width, leaving the height alone.
    ///
    /// - Parameters:
    ///   - usePixelAspectRatio: Compute the dimensions maintaining
pixel aspect
    ///     ratio.
    ///   - useCleanAperture: Compute the dimensions using the clean
aperture.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func presentationDimensions
Bool   true                     Bool   true     CGSize

    /// Returns the clean aperture.
    ///
    /// The clean aperture is a rectangle that defines the portion of the encoded
    /// pixel dimensions that represents image data valid for display.
    ///
    /// - Parameter originIsAtTopLeft: Pass `true` if the CGRect will
be used in
    ///   an environment where `(0, 0)` is at the top-left corner of an
enclosing
    ///   rectangle and y coordinates increase as you go down. Pass `false`
if the
    ///   `CGRect` will be used in an environment where `(0, 0)` is at the
    ///   bottom-left corner of an enclosing rectangle and y coordinates increase
    ///   as you go up.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func cleanAperture                            Bool
CGRect

    /// Keys that are used both as `CMVideoFormatDescription`
extensions and
    /// CVImageBuffer attachments and attributes.
    ///
    /// When specifying a format description for a `CMSampleBuffer`, the
format
    /// description must be consistent with formatting information attached to the
    /// `CVImageBuffer`. The width, height, and codec type must match (for
    /// `CVPixelBuffers` the codec type is given by
    /// `CVPixelBufferGetPixelFormatType(pixelBuffer)`; for other
`CVImageBuffers`,
    /// the codec type must be `0`).
    ///
    /// The format description extensions must match the image buffer
attachments
    /// for all the keys in the list returned by this function (if absent in
    /// either they must be absent in both).
    ///
    /// See
`CMVideoFormatDescriptionGetExtensionKeysCommonWithImageBuffer
s`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public static var extensionKeysCommonWithImageBuffers
 CMFormatDescription Extensions Key     get

    /// Checks to see if the format description matches an image buffer.
    ///
    /// This function uses the keys returned by
    /// `extensionKeysCommonWithImageBuffers` to compares the
extensions of the
    /// format description to the attachments of the given image buffer (if an
    /// attachment is absent in either it must be absent in both).
    ///
    /// It also checks `kCMFormatDescriptionExtension_BytesPerRow`
against
    /// `CVPixelBufferGetBytesPerRow`, if applicable.
    ///
    /// - Parameter imageBuffer: Image buffer validate against.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func matchesImageBuffer _
CVImageBuffer     Bool


@available macOS 14.0  iOS 17.0  tvOS 17.0  watchOS 10.0
visionOS 1.0
extension CMFormatDescription

    /// Returns the tag collections.
    ///
    /// This property provides the VideoLayerIDs and LeftAndRightViewIDs from
hvcC and 3D Reference
    /// Displays Info SEI in the formatDescription. The returned values can be
used to enable the
    /// multi-image decoding with
kVTDecompressionPropertyKey_RequestedMVHEVCVideoLayerIDs.
    /// It also gives the eye mapping information for the pixel buffers of the
decoded CMTaggedBuffers.
    @available macOS 14.0  iOS 17.0  tvOS 17.0  watchOS 10.0
visionOS 1.0
    public var tagCollections    CMTag       get


@available macOS 14.0  iOS 17.0  tvOS 17.0  watchOS 10.0
visionOS 1.0
extension CMFormatDescription
```

```
    ///  Checks to see if the format description matches the provided tagged
buffers.
    ///
    ///  This function returns true if the format description matches
    ///  the format description in the specified tagged buffers, false otherwise.
    ///
    /// - Parameter taggedBuffers:  Tagged buffers to validate against.
    @available macOS 14.0  iOS 17.0  tvOS 17.0  watchOS 10.0
visionOS 1.0
    public func matchesTaggedBufferGroup _
 CMTaggedBuffer      Bool


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMFormatDescription

    ///  Returns the display flags.
    ///
    ///  These are the flags that control how the text appears. The function can
    ///  throw `CMFormatDescription.Error.valueNotAvailable` for
formats without
    ///  display flags.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func displayFlags   throws
CMFormatDescription Extensions Value TextDisplayFlags

    ///  Returns horizontal and vertical justification.
    ///
    ///  Values are `TextJustification` constants. The function throws
    ///  `CMFormatDescription.Error.valueNotAvailable` for format
descriptions that
    ///  do not carry text justification.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func justification   throws
CMFormatDescription Extensions Value TextJustification

CMFormatDescription Extensions Value TextJustification

    ///  Returns the default text box.
    ///
    ///  Within a text track, text is rendered within a text box. There is a
    ///  default text box set, which can be over-ridden by a sample. The function
    ///  can throw `CMFormatDescription.Error.valueNotAvailable`
for format
    ///  descriptions that do not carry a default text box.
    ///
    /// - Parameters:
    ///   - originIsAtTopLeft: Pass `true` if the `CGRect` will be
used in an
    ///     environment where `(0, 0)` is at the top-left corner of an
enclosing
    ///     rectangle and y coordinates increase as you go down. Pass
`false` if
    ///     the `CGRect` will be used in an environment where `(0, 0)` is
at the
    ///     bottom-left corner of an enclosing rectangle and y coordinates
    ///     increase as you go up.
    ///   - heightOfTextTrack: If `originIsAtTopLeft` is `false`,
pass the height
    ///     of the enclosing text track or destination. This value will be used to
    ///     properly compute the default text box for the given origin. Ignored if
    ///     `originIsAtTopLeft` is `true`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func defaultTextBox                    Bool
                       CGFloat  throws    CGRect

    /// Returns the default style.
    ///
    /// The function throws
`CMFormatDescription.Error.valueNotAvailable` for
    /// format descriptions that do not carry default style information.
    /// - Returns:
    ///   - localFontID: Font number, local to the format description.
    ///   - bold: `true` if style includes Bold.
    ///   - italic: `true` if style includes Italic.
    ///   - underline: `true` if style includes Underline.
    ///   - fontSize: font size in points.
    ///   - colocComponents: Components are in order red, green, blue, alpha.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func defaultStyle   throws                   Int
      Bool            Bool            Bool            CGFloat
                     CGFloat

    /// Returns the font name for a local font ID.
    ///
    /// Some format descriptions carry a mapping from local font IDs to font
    /// names. The function returns
`CMFormatDescription.Error.valueNotAvailable`
    /// for format descriptions that do not carry such a font mapping table.
    ///
    /// - Parameter localFontID: Font number, local to the format
description.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
```

```
    public func fontName                Int   throws    String


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMFormatDescription

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct TimeCode   Sendable

        ///  TimeCode Flags
        ///
        ///  Flags passed to
`init(timeCodeFormatType:frameDuration:frameQuanta:flags:exten
sions:)`.
        @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS
6.0  visionOS 1.0
        public struct Flag   OptionSet  Sendable

            ///  The corresponding value of the raw type.
            ///
            ///  A new instance initialized with `rawValue` will be equivalent
to this
            ///  instance. For example:
            ///
            ///      enum PaperSize: String {
            ///          case A4, A5, Letter, Legal
            ///      }
            ///
            ///      let selectedSize = PaperSize.Letter
            ///      print(selectedSize.rawValue)
            ///      // Prints "Letter"
            ///
            ///      print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue)!)
            ///      // Prints "true"
            public let rawValue  UInt32

            ///  Creates a new option set from the given raw value.
            ///
            ///  This initializer always succeeds, even if the value passed as
`rawValue`
            ///  exceeds the static properties declared as part of the option set.
This
            ///  example creates an instance of `ShippingOptions` with a
raw value beyond
            ///  the highest element, with a bit mask that effectively contains all
the
            ///  declared static members.
            ///
            ///     let extraOptions =
ShippingOptions(rawValue: 255)
            ///
print(extraOptions.isStrictSuperset(of: .all))
            ///     // Prints "true"
            ///
            /// - Parameter rawValue: The raw value of the option set
to create. Each bit
            ///   of `rawValue` potentially represents an element of the
option set,
            ///   though raw values may include bits that are not defined as
distinct
            ///   values of the `OptionSet` type.
            public init           UInt32

            /// Timecodes are to be rendered in drop-frame format.
            public static let dropFrame
CMFormatDescription TimeCode Flag

            /// Timecode rolls over every 24 hours.
            public static let twentyFourHourMax
CMFormatDescription TimeCode Flag

            /// Track may contain negative timecodes.
            public static let negTimesOK
CMFormatDescription TimeCode Flag

            /// The type of the elements of an array literal.
            @available iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0  macOS 10.15
            public typealias ArrayLiteralElement
CMFormatDescription TimeCode Flag

            /// The element type of the option set.
            ///
            /// To inherit all the default implementations from the
`OptionSet` protocol,
            /// the `Element` type must be `Self`, the default.
            @available iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0  macOS 10.15
            public typealias Element
CMFormatDescription TimeCode Flag

            /// The raw type that can be used to represent all values of the
conforming
            /// type.
            ///
            /// Every distinct value of the conforming type has a
corresponding unique
```

```
            ///  value of the `RawValue` type, but there may be values of the
`RawValue`
            ///  type that don't have a corresponding value of the conforming
type.
            @available iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0  macOS 10.15
            public typealias RawValue   UInt32



    ///  The duration of each frame (eg. `100/2997`).
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var frameDuration  CMTime    get

    ///  The frames/sec for timecode (eg. `30`) OR frames/tick for counter mode.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var frameQuanta  UInt32    get

    ///  The flags for `.dropFrame`, `._24HourMax`, `.negTimesOK`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var timeCodeFlags
CMFormatDescription TimeCode Flag    get


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMFormatDescription

    ///  Returns the key associated with the metadata for the given local ID.
    ///
    /// - Parameter localKeyID:  Local ID identifying the key associated
with the
    ///     metadata description.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func keyWithLocalID _            OSType
 String   CFPropertyList

    ///  An array of metadata identifiers.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var identifiers   String    get


@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
extension CMTime
```

```
    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public init          Double
CMTimeScale

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public init        CMTimeValue             CMTimeScale


@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
extension CMTime

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public var isValid  Bool   get

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public var isPositiveInfinity  Bool   get

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public var isNegativeInfinity  Bool   get

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public var isIndefinite  Bool   get

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public var isNumeric  Bool   get

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public var hasBeenRounded  Bool   get

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public var seconds  Double   get

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public func convertScale _               Int32
CMTimeRoundingMethod     CMTime
```

```
@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
extension CMTime

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public static func             CMTime          CMTime
CMTime

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public static func             CMTime             CMTime
   CMTime


@available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
extension CMTime   Equatable  Comparable

    /// Returns a Boolean value indicating whether the value of the first
    /// argument is less than that of the second argument.
    ///
    /// This function is the only requirement of the `Comparable` protocol. The
    /// remainder of the relational operator functions are implemented by the
    /// standard library for any type that conforms to `Comparable`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public static func           CMTime         CMTime
Bool

    /// Returns a Boolean value indicating whether the value of the first
    /// argument is less than or equal to that of the second argument.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public static func            CMTime         CMTime
Bool

    /// Returns a Boolean value indicating whether the value of the first
    /// argument is greater than that of the second argument.
    ///
    /// - Parameters:
```

```
    ///    - lhs: A value to compare.
    ///    - rhs: Another value to compare.
    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public static func            CMTime          CMTime
Bool

    /// Returns a Boolean value indicating whether the value of the first
    /// argument is greater than or equal to that of the second argument.
    ///
    /// - Parameters:
    ///    - lhs: A value to compare.
    ///    - rhs: Another value to compare.
    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public static func             CMTime          CMTime
Bool

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///    - lhs: A value to compare.
    ///    - rhs: Another value to compare.
    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public static func             CMTime          CMTime
Bool

    @available macOS 10.7  iOS 4.0  tvOS 9.0  watchOS 6.0
visionOS 1.0
    public static func             CMTime          CMTime
Bool


@available macOS 13.0  iOS 16.0  tvOS 16.0  watchOS 9.0
visionOS 1.0
extension CMTime   Hashable

    /// Hashes the essential components of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform to the `Hashable` protocol. The
    /// components used for hashing must be the same as the components
compared
    /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
```

```
    /// with each of these components.
    ///
    /// - Important: In your implementation of `hash(into:)`,
    ///     don't call `finalize()` on the `hasher` instance provided,
    ///     or replace it with a different instance.
    ///     Doing so may become a compile-time error in the future.
    ///
    /// - Parameter hasher: The hasher to use when combining the
components
    ///     of this instance.
    @available macOS 13.0  iOS 16.0  tvOS 16.0  watchOS 9.0
visionOS 1.0
    public func hash              inout Hasher

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable`
requirement. To
    ///     conform to `Hashable`, implement the `hash(into:)` requirement
instead.
    ///     The compiler provides an implementation for `hashValue` for you.
    public var hashValue  Int   get


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMBlockBuffer

    /// A slice of a `CMBlockBuffer` instance.
    ///
    /// - Important: Long-term storage of `CMBlockBuffer.Slice`
instances is
    ///     discouraged. A slice holds a reference to the entire storage of a larger
    ///     block buffer, not just to the portion it presents, even after the original
    ///     buffer's lifetime ends. Long-term storage of a slice may therefore
prolong
    ///     the lifetime of bytes that are no longer otherwise accessible, which can
    ///     appear to be memory and object leakage.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct Slice



@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMBlockBuffer   CMBlockBufferProtocol
```

```
    /// `CMBlockBuffer` instance to operate on.
    public var owner  CMBlockBuffer   get

    /// The position of the first element.
    public var startIndex  Int   get

    /// The "past the end" position.
    public var endIndex  Int   get


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMBlockBuffer

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public typealias CustomBlockAllocator    Int
UnsafeMutableRawPointer

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public typealias CustomBlockDeallocator
 UnsafeMutableRawPointer  Int     Void


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMBlockBuffer

    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct Error   Sendable

        public static let structureAllocationFailed  NSError

        public static let blockAllocationFailed  NSError

        public static let badCustomBlockSource  NSError

        public static let badOffsetParameter  NSError

        public static let badLengthParameter  NSError

        public static let badPointerParameter  NSError

        public static let emptyBlockBuffer  NSError

        public static let unallocatedBlock  NSError
        public static let insufficientSpace  NSError



@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMBlockBuffer

    /// Flags controlling behaviors and features of `CMBlockBuffer` APIs.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public struct Flags   OptionSet  Sendable

        /// The corresponding value of the raw type.
        ///
        /// A new instance initialized with `rawValue` will be equivalent to this
        /// instance. For example:
        ///
        ///     enum PaperSize: String {
        ///         case A4, A5, Letter, Legal
        ///     }
        ///
        ///     let selectedSize = PaperSize.Letter
        ///     print(selectedSize.rawValue)
        ///     // Prints "Letter"
        ///
        ///     print(selectedSize == PaperSize(rawValue:
selectedSize.rawValue)!)
        ///     // Prints "true"
        public let rawValue  UInt32

        /// Creates a new option set from the given raw value.
        ///
        /// This initializer always succeeds, even if the value passed as
`rawValue`
        /// exceeds the static properties declared as part of the option set. This
        /// example creates an instance of `ShippingOptions` with a raw
value beyond
        /// the highest element, with a bit mask that effectively contains all the
        /// declared static members.
        ///
        ///     let extraOptions = ShippingOptions(rawValue:
255)
        ///     print(extraOptions.isStrictSuperset(of: .all))
        ///     // Prints "true"
        ///
        /// - Parameter rawValue: The raw value of the option set to
create. Each bit
        ///   of `rawValue` potentially represents an element of the option
```

```
set,
        ///     though raw values may include bits that are not defined as distinct
        ///     values of the `OptionSet` type.
        public init           UInt32

        ///  When passed to routines that accept block allocators, causes the
memory
        ///  block to be allocated immediately.
        public static let assureMemoryNow  CMBlockBuffer Flags

        ///  Used with `makeContiguous()` to cause it to always produce an
allocated
        ///  copy of the desired data.
        public static let alwaysCopyData  CMBlockBuffer Flags

        ///  Passed to `append(bufferReference:flags:)` and
        ///  `init(bufferReference:flags:)` to suppress reference
depth optimization.
        public static let dontOptimizeDepth
CMBlockBuffer Flags

        ///  Passed to `append(bufferReference:flags:)` and
        ///  `init(bufferReference:flags:)` to allow references into a
        ///  `CMBlockBuffer` that may not yet be populated.
        public static let permitEmptyReference
CMBlockBuffer Flags

        ///  The type of the elements of an array literal.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias ArrayLiteralElement
CMBlockBuffer Flags

        ///  The element type of the option set.
        ///
        ///  To inherit all the default implementations from the `OptionSet`
protocol,
        ///  the `Element` type must be `Self`, the default.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias Element   CMBlockBuffer Flags

        ///  The raw type that can be used to represent all values of the
conforming
        ///  type.
        ///
        ///  Every distinct value of the conforming type has a corresponding
unique
        ///  value of the `RawValue` type, but there may be values of the
`RawValue`
```

```
        /// type that don't have a corresponding value of the conforming type.
        @available iOS 13.0  tvOS 13.0  watchOS 6.0  visionOS
1.0  macOS 10.15
        public typealias RawValue   UInt32



@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMBlockBuffer

    /// Adds a memory block, to allocate with a `CFAllocator`.
    ///
    /// Adds a memory block to an existing `CMBlockBuffer`. The
`CMBlockBuffer`'s
    /// total data length will be increased by the specified `range` length.
    ///
    /// If `.assureMemoryNow` is set in the `flags` parameter, the memory
block is
    /// allocated immediately using the `allocator`.
    ///
    /// Note that append operations are not thread safe, so care must be taken
    /// when appending to `CMBlockBuffer`s that are used by multiple
threads.
    ///
    /// - Parameters:
    ///   - length:  Overall length of the memory block in bytes. Must not be
zero.
    ///       This is the size to allocate when `assureBlockMemory()` is
called.
    ///   - allocator:  Allocator to be used for allocating the memory block.
    ///   - range:  Range within the memory block to which the
`CMBlockBuffer`
    ///       should refer to data. If `nil`, the whole memory block is used.
    ///   - flags:  Feature and control flags.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func append         Int                CFAllocator
                                  Range Int       nil
CMBlockBuffer Flags       throws

    /// Adds a memory block, alreay allocated with a `CFAllocator`.
    ///
    /// Adds a memory block to an existing `CMBlockBuffer`. The
`CMBlockBuffer`'s
    /// total data length will be increased by the specified `buffer` length.
    ///
    /// Note that append operations are not thread safe, so care must be taken
    /// when appending to `CMBlockBuffer`s that are used by multiple
threads.
```

```
    ///
    /// - Parameters:
    ///   - buffer:  Block of memory to hold buffered data. The block will be
used
    ///       and will be deallocated when the `CMBlockBuffer` is finalized
(i.e.
    ///       released for the last time).
    ///   - allocator:  Allocator to be used for deallocating the `buffer`.
Pass
    ///      `kCFAllocatorNull` if no deallocation is desired.
    ///   - flags:  Feature and control flags.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func append         UnsafeMutableRawBufferPointer
           CFAllocator
CMBlockBuffer Flags       throws

    /// Adds a sliced memory block, alreay allocated with a `CFAllocator`.
    ///
    /// Adds a memory block to an existing `CMBlockBuffer`. The
`CMBlockBuffer`'s
    /// total data length will be increased by the specified `buffer` slice
    /// length.
    ///
    /// Note that append operations are not thread safe, so care must be taken
    /// when appending to `CMBlockBuffer`s that are used by multiple
threads.
    ///
    /// - Parameters:
    ///   - buffer:  Slice in a block of memory to hold buffered data. The
block
    ///       will be used and will be deallocated when the `CMBlockBuffer` is
    ///       finalized (i.e. released for the last time). The `CMBlockBuffer`
will
    ///       refer to the data in the `buffer` slice.
    ///   - allocator:  Allocator to be used for deallocating the `buffer`.
Pass
    ///      `kCFAllocatorNull` if no deallocation is desired.
    ///   - flags:  Feature and control flags.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func append
Slice UnsafeMutableRawBufferPointer              CFAllocator
                              CMBlockBuffer Flags       throws

    /// Adds a memory block, to allocate with a custom allocator.
    ///
    /// Adds a memory block to an existing `CMBlockBuffer`. The
`CMBlockBuffer`'s
    /// total data length will be increased by the specified `range` length.
    ///
    /// If `.assureMemoryNow` is set in the `flags` parameter, the memory
block is
    /// allocated immediately using the `allocator`.
    ///
    /// Note that append operations are not thread safe, so care must be taken
    /// when appending to `CMBlockBuffer`s that are used by multiple
threads.
    ///
    /// - Parameters:
    ///   - length: Overall length of the memory block in bytes. Must not be
zero.
    ///       This is the size to allocate when `assureBlockMemory()` is
called.
    ///   - allocator: Allocator to be used for allocating the memory block.
    ///   - deallocator: Deallocator to be used for deallocating the
memory block.
    ///   - range: Range within the memory block to which the
`CMBlockBuffer`
    ///       should refer to data. If `nil`, the whole memory block is used.
    ///   - flags: Feature and control flags.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func append         Int              @escaping
CMBlockBuffer CustomBlockAllocator                @escaping
CMBlockBuffer CustomBlockDeallocator         Range Int
nil          CMBlockBuffer Flags        throws

    /// Adds a memory block, alreay allocated with a custom allocator.
    ///
    /// Adds a memory block to an existing `CMBlockBuffer`. The
`CMBlockBuffer`'s
    /// total data length will be increased by the specified `buffer` length.
    ///
    /// Note that append operations are not thread safe, so care must be taken
    /// when appending to `CMBlockBuffer`s that are used by multiple
threads.
    ///
    /// - Parameters:
    ///   - buffer: Block of memory to hold buffered data. The block will be
used
    ///       and will be deallocated when the new CMBlockBuffer is finalized (i.e.
    ///       released for the last time).
    ///   - deallocator: Deallocator to be used for deallocating the buffer.
    ///   - flags: Feature and control flags.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func append         UnsafeMutableRawBufferPointer
            @escaping CMBlockBuffer CustomBlockDeallocator
       CMBlockBuffer Flags       throws
    /// Adds a sliced memory block, alreay allocated with a custom allocator.
    ///
    /// Adds a memory block to an existing `CMBlockBuffer`. The
`CMBlockBuffer`'s
    /// total data length will be increased by the specified `buffer` slice
    /// length.
    ///
    /// Note that append operations are not thread safe, so care must be taken
    /// when appending to `CMBlockBuffer`s that are used by multiple
threads.
    ///
    /// - Parameters:
    ///   - buffer: Slice in a block of memory to hold buffered data. The
block
    ///       will be used and will be deallocated when the new
`CMBlockBuffer` is
    ///       finalized (i.e. released for the last time). The `CMBlockBuffer`
will
    ///       refer to the data in the slice.
    ///   - deallocator: Deallocator to be used for deallocating the buffer.
    ///   - flags: Feature and control flags.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func append
Slice UnsafeMutableRawBufferPointer                @escaping
CMBlockBuffer CustomBlockDeallocator
CMBlockBuffer Flags          throws

    /// Adds a `CMBlockBuffer` reference.
    ///
    /// Adds a buffer reference to (a possibly subset portion of) another
    /// `CMBlockBuffer` to an existing `CMBlockBuffer`. The
`CMBlockBuffer`'s
    /// total data length will be increased by the specified `bufferReference`
    /// length.
    ///
    /// Note that append operations are not thread safe, so care must be taken
    /// when appending to `CMBlockBuffer`s that are used by multiple
threads.
    ///
    /// - Parameters:
    ///   - bufferReference: Slice of a `CMBlockBuffer` to refer to.
Unless
    ///     `.permitEmptyReference` is passed, it must not be empty.
    ///   - flags: Feature and control flags.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func append T                         T
CMBlockBuffer Flags          throws where T
```

```
CMBlockBufferProtocol

    ///  Assures all memory blocks are allocated.
    ///
    ///  Traverses the possibly complex `CMBlockBuffer`, allocating the
memory for
    ///  any constituent memory blocks that are not yet allocated.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func assureBlockMemory   throws


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMBlockBuffer

    ///  Gains access to the data represented by a `CMBlockBuffer`.
    ///
    ///  Gains access to the data represented by a `CMBlockBuffer`. A
mutable
    ///  buffer pointer into a memory block is given to the closure which
    ///  corresponds to the offset within the `CMBlockBuffer`. This buffer may
be
    ///  smaller than the number of bytes actually available starting at the offset
    ///  if the `dataLength` of the `CMBlockBuffer` is covered by multiple
memory
    ///  blocks (a noncontiguous `CMBlockBuffer`). The buffer pointer will
remain
    ///  valid as long as the original `CMBlockBuffer` is referenced - once the
    ///  `CMBlockBuffer` is released for the last time, any buffer pointers into it
    ///  will be invalid.
    ///
    /// - Parameters:
    ///    - offset:  Offset within the buffer's offset range.
    ///    - body:  A closure with an `UnsafeMutableRawBufferPointer`
parameter that
    ///        points to contiguous storage in the block buffer.
    /// - Returns:  The return value, if any, of the body closure parameter.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public func withUnsafeMutableBytes R                  Int
  0  _        UnsafeMutableRawBufferPointer  throws    R
throws    R


@available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
extension CMBlockBuffer

    ///  Indicates whether the `CMBlockBuffer` is empty.
    ///
    /// Indicates whether the `CMBlockBuffer` is empty, i.e., devoid of any
    /// memory blocks or `CMBlockBuffer` references. Note that a
`CMBlockBuffer`
    /// containing a not-yet allocated memory block is not considered empty.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public var isEmpty  Bool   get

    /// The `CFTypeID` corresponding to `CMBlockBuffer`.
    @available macOS 10.15  iOS 13.0  tvOS 13.0  watchOS 6.0
visionOS 1.0
    public class var typeID  CFTypeID   get
```