iOS API

version 1.38

Revision History

Date	Version	Description	Author
11/18/2013	1.38	Updated AirPlay with Image	Tony Seaward
08/20/2013	1.37	Added AirPlay capability for encrypted loopback	Tony Seaward
07/25/2013	1.36	Updated Version Requirement for JSON	Tony Seaward
06/05/2013	1.35	Updated Requirement for JSON Fixed Requirement for Initialize Guideline	Tony Seaward
04/24/2013	1.34	Added JSON reference	Tony Seaward
04/04/2013	1.33	Clarified Bits vs Bytes in Event_Bitrates	Tony Seaward
04/04/2013	1.32	Included Required Libs to upgrade from 5.0 to 6.0	Tony Seaward
04/01/2013	1.31	Updated Images, Descriptions for WV_Initialize and WV_Terminate	Tony Seaward
03/31/2013	1.30	Formatting Changes	Alex Lee
03/26/2013	1.29	Restructured Document for improved reading flow	Tony Seaward

See Full Revision History

Contents

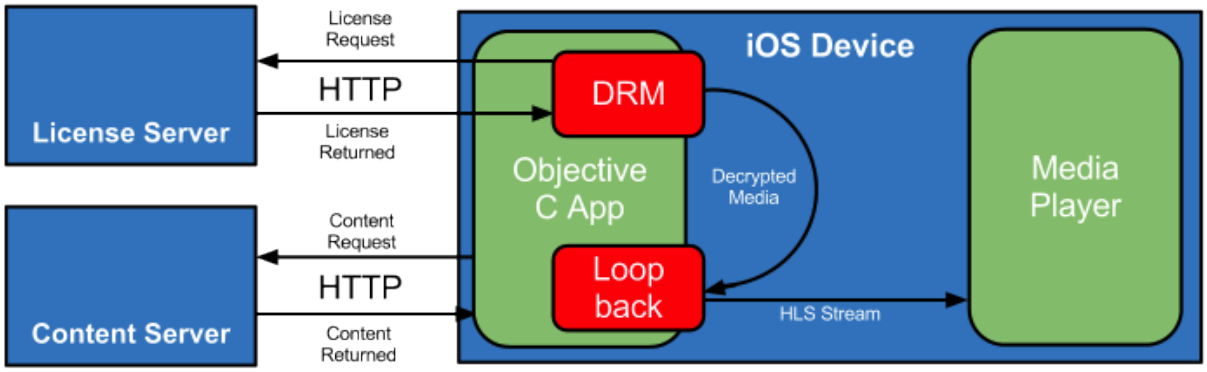
Purpose

This document defines the interface for licensing, decryption, and playback of Widevine-encrypted media on Apple iOS devices.

This document is targeted to application developers who want to play content to iOS devices with Widevine-encrypted media.

Playback Overview

The Widevine iOS client provides playback over a HTTP loopback interface. iOS does not allow multiple applications to run simultaneously, so the only application that can stream from the loopback interface is the media player linked to the Widevine client.



Playback API

The WV_Play function is used to initiate playback for a local or remote asset. WV_Play will use the relative file path within the application for a local asset or the HTTP/HTTPS URL for a remote asset. Typically this return is passed directly to the Apple movie player object MPMoviePlayer or AVPlayer. ***The asset must contain the correct file extension*** - either .wvm, .mp4 or .smil.

If playback of a non-registered remote asset is initiated, the Widevine iOS Client sends a license request but begins playback without delay.

Normally, assets have a short (10 second clear lead) initial non-encrypted segment to allow the license request to occur in the background to help avoid any interruption during playback, unless a license error (EMMFailed) is returned.

HTTP Streaming

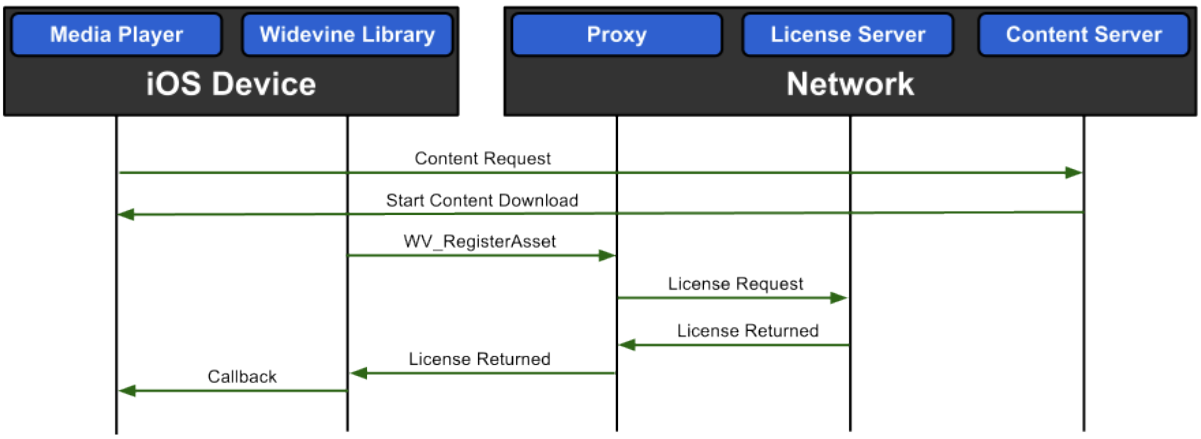
HTTP streaming does not require registering or downloading of the asset; however, streaming via Apple's MPMoviePlayer will make a request for the entire file in order to identify the available bitrates within the file and then switch to the appropriate stream based on the network connection.

Remote Assets

The Widevine iOS client supports the use of streamed content to the iOS device. Streaming playback allows for closed-range byte requests via the HTTP GET method.

Registering a remote asset prior to playback is optional:

- Streaming assets will automatically invoke a license request (if no license exists) when content is being played.
- Streaming assets may also be played while simultaneously acquiring a license via WV_RegisterAsset - which stores the license locally.
    - Persistent license acquisition is performed using WV_RegisterAsset which can occur with or without content playback.



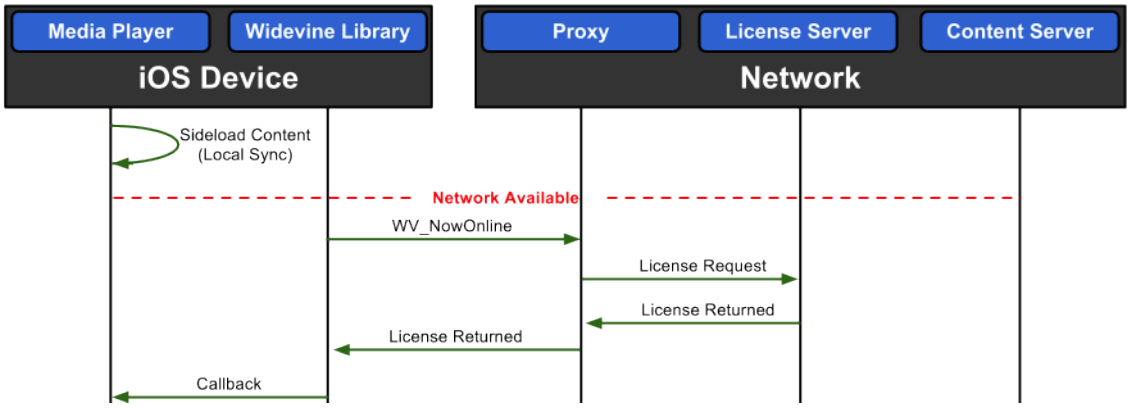
Local Playback

Locally stored (downloaded) assets must be registered prior to offline playback with a license that exists on the device. With an active network connection, a license request must be made to the Widevine license server by calling WV_RegisterAsset. The Widevine library stores the license securely and allows for querying of the available rights associated with the asset while offline.

Offline Assets

If content was previously synchronized to the device and there is no network connection and no license exists on the device for the asset, WV_NowOnline must be called when the network becomes available to obtain the license required for decryption.

- WV_NowOnline is equivalent to a license refresh.



Playback Flow

1. Create callback to listen for Widevine Events
2. Initialize Widevine Settings
3. Register the Asset (invokes License Retrieval)
4. Check API Return Status
5. Play Content

WViOsApiStatusCallback

```
typedef WViOsApiStatus(*WViOsApiStatusCallback)(WViOsApiEvent event,
                                                NSDictionary *attributes);
```

The Widevine Library will use this callback to obtain the status and/or keycode and value for a Widevine iOS operation, event, or asset.

These attribute keys are coded in the NSDictionary object:

Key	NS Dictionary Description	WViOsApiEvent(s)
`WVEMMTimeRemainingKey`	NSNumber - Seconds remaining for this license. Once exceeded, the license is invalid.	EMMReceived, Playing, QueryStatus
`WVPurchaseTimeRemainingKey`	NSNumber – Seconds remaining on purchase duration. Once it reaches 0, no more licenses can be requested for the asset.	EMMReceived, Playing, QueryStatus, Registered
`WVDistributionTimeRemainingKey`	NSNumber - Seconds remaining for this asset's distribution. Once exceeded, this asset will no longer decrypt, even with a valid license.	EMMReceived, Playing, QueryStatus, Registered
`WVTimeSinceLastPlaybackKey`	NSNumber – Number of seconds since last playback.	EMMReceived, Playing, QueryStatus, Registered
`WVAssetPathKey`	NSString – Relative path to the asset on the device.	EMMReceived, EMMRemoved, EMMFailed, Playing, PlayFailed, QueryStatus, Registered, Unregistered, Stopped
`WVIsEncryptedKey`	NSNumber – Boolean indicating whether the asset is a Widevine encrypted file.	EMMReceived, Playing, QueryStatus, Registered
`WViOsApiStatusKey`	A WViOsApiStatus enum value, encoded as NSNumber.	Initialize, InitializeFailed, EMMFailed, PlayFailed, QueryStatus, Unregistered, EMMRemoved
`WVVersionKey`	NSString - Version of the Widevine software	PlayFailed, EMMFailed, InitializeFailed, Initialized
`WVAssetTypeKey`	NSNumber - 0 = Unknown, 1 = file 2 = HTTP stream	EMMReceived, EMMRemoved, EMMFailed, Playing, WPlayFailed, QueryStatus, Registered, Unregistered, Stopped

`WVSystemIDKey`	NSNumber – Value of CA system ID	EMMReceived, EMMFailed, Playing, PlayFailed, QueryStatus, Registered
`WVAssetIDKey`	NSNumber – Value of CA asset ID.	EMMReceived, EMMFailed, Playing, PlayFailed, QueryStatus, Registered
`WVKeyIDKey`	NSNumber – Value of CA key ID.	EMMReceived, EMMFailed, Playing, PlayFailed, QueryStatus, Registered
`WVErrorKey`	NSNumber (error code), or NSString (error description). Contains request specific codes, such as licensing errors.	EMMFailed
`WVCopyControlInformation_EMIKey`	NSString - “NoRestrictions”, “NoFurtherCopying”, “OneGenerationCopyPermitted”, “NoCopy”	EMMReceived, Playing, QueryStatus, Registered
`WVCopyControlInformation_APSKey`	NSString - “NoRestrictions”, “AGCOn”, “ACGOn2” “AGCOn4”	EMMReceived, Playing, QueryStatus, Registered
`WVCopyControlInformation_CITKey`	NSNumber - 0 or 1	EMMReceived, Playing, QueryStatus, Registered
`WVCopyControlInformation_HDCPKey`	NSNumber - 0 or 1	EMMReceived, Playing, QueryStatus, Registered

WV_Settings

WV_Initialize is called with an NSDictionary of name/value pairs.

The following keys are recognized:

Key	Description
`WVDRMServerKey`	NSString. Required. URL for the CA server CGI. If specified, it overrides the URL coded into the assets (for encrypted .mp4 only).
`WVAssetDBPathKey`	NSString. Optional. File system path of the directory where the asset database should be stored. If specified, it overrides the default (user's Application Support directory).
`WVDRMAckUrlKey`	NSString. Optional. URL for the CA server CGI to respond when an EMM is received.
`WVAssetRootKey`	NSString. Optional. File system path of the directory where the assets are stored. All asset paths are relative to this "root" path. This is required to override the default path (sandbox) that is reset whenever the iPhone App is updated. If specified, this path overrides the default (user's Documents directory).
`WVStreamIdKey`	NSString. Optional. Contains Stream ID to be passed to CA system.
`WVDeviceIdKey`	NSString. Optional. Contains Device ID to be passed to CA system. Recommend using WV_GetDeviceId to set value.
`WVClientIPKey`	NSString. Optional. Sets the client IP address. Default is system IP address.

`WVPlayerDrivenAdaptationKey`	NSString. Optional. Values = 1 or 0. Switches between Apple Native Player Adaption (1) and Widevine Adaption (0). 1 = Enables player-driven adaptation. 0 = Disables player-driven adaptation. Default value = 1 (player-driven adaptation is recommended) Player-driven adaptation can be disabled if: ● All tracks (bit rates) have 100% consistent GOPs; that is, the time index is identical for all I-frames. ● All tracks are 100% consistent in terms of audio and video encoding parameters, including video resolution, video frame rate, audio bit rate.
`WVCAUserDataKey`	NSString. Optional. The userdata to be passed to the licensing system.
`WVPreloadTimeoutKey`	NSString. Optional. The number of milliseconds to attempt a preload before failure.
`WVSessionIdKey`	NSString. Deprecated. *Note: not deprecated in the header.*
`WVPortalKey`	NSString. Required. A Widevine-assigned value.
`WVUseJSONKey`	Boolean. Optional. Determines which license request format type to use. 1 = Use JSON format 0 = Use HTTP POST format (default if not specified)
`WVUseEncryptedLoopback`	Boolean. Optional. Enables AirPlay support by securing the AirPlay stream. 1 = Enable encypted loopback 0 = Disable encrypted loopback (default if not specified) This parameter requires `WVPlayerDrivenAdaptationKey` enabled.

NOTE: Be sure to set the values in the dictionary that is passed in before calling WV_Initialize.

WViOsApiStatus

WViOsApiStatus is the result code for many Widevine functions to indicate the success (0) or failure type (1 to 20). The WViOsApiStatus code is returned by a WViOsApiStatusCallback .

```
typedef enum WViOsApiStatus {
      WViOsApiStatus_OK                          = 0,
      WViOsApiStatus_NotInitialized              = 1,
      WViOsApiStatus_AlreadyInitialized          = 2,
      WViOsApiStatus_CantConnectToMediaServer    = 3,
      WViOsApiStatus_BadMedia                    = 4,
      WViOsApiStatus_CantConnectToDrmServer      = 5,
      WViOsApiStatus_NotEntitled                 = 6,
      WViOsApiStatus_licenseDenied               = 7,
      WViOsApiStatus_LostConnection              = 8,
      WViOsApiStatus_EntitlementExpired          = 9,
      WViOsApiStatus_NotEntitledByReg            = 10,
      WViOsApiStatus_BadUrl                      = 11,
      WViOsApiStatus_FileNotPresent              = 12,
      WViOsApiStatus_NotRegistered               = 13,
      WViOsApiStatus_AlreadyRegistered           = 14,
      WViOsApiStatus_NotPlaying                  = 15,
      WViOsApiStatus_AlreadyPlaying              = 16,
      WViOsApiStatus_FileSystemError             = 17,
      WViOsApiStatus_AssetDBWasCorrupted         = 18,
      WViOsApiStatus_JailBreakDectected          = 19,
      WViOsApiStatus_UnknownError                = 20,
} WViOsApiStatus;
```

WViOsApiEvent

WViOsApiEvent is an enumerated event identifier. Event and Status can be sent to the iOS Media player using WViOsApiStatusCallback.

```
typedef enum WViOsApiEvent {
      WViOsApiEvent_NullEvent                    = 0,
      WViOsApiEvent_EMMReceived                  = 1,
      WViOsApiEvent_EMMFailed                    = 2,
      WViOsApiEvent_Playing                      = 3,
      WViOsApiEvent_PlayFailed                   = 4,
      WViOsApiEvent_Stopped                      = 5,
      WViOsApiEvent_QueryStatus                  = 6,
      WViOsApiEvent_EndOfList                    = 7,
      WViOsApiEvent_Initialized                  = 8,
      WViOsApiEvent_InitializeFailed             = 9,
      WViOsApiEvent_Terminated                   = 10,
      WViOsApiEvent_EMMRemoved                   = 11,
      WViOsApiEvent_Registered                   = 12,
      WViOsApiEvent_Unregistered                 = 13,
      WViOsApiEvent_SetCurrentBitrate            = 14,
      WViOsApiEvent_Bitrates                     = 15,
      WViOsApiEvent_StoppingOnError              = 16,
      WViOsApiEvent_VideoParams                  = 17,
      WViOsApiEvent_AudioParams                  = 18,
      WViOsApiEvent_Subtitles                    = 19,
      WViOsApiEvent_AudioOnlyTracks              = 20,
} WViOsApiEvent;
```

WV Methods

WV_Initialize

```
WViOsApiStatus WV_Initialize(const WViOsApiStatusCallback callback,
                             NSDictionary *settings);
```

The WV_Initialize method is required to begin a session. Other methods should be called between WV_Initialize and WV_Terminate to function properly. WV_Initialize is only required to be called again after WV_Terminate has been called.

Parameters

[in] callback - Specifies callback function in client for information about assets.

[in] settings – name/value pairs for various settings, see WV_Settings.

Returns

WViOsApiStatus_OK on success, otherwise a WViOsApiStatus error.

Events

WViOsApiEvent_Initialized, WViOsApiEvent_InitializeFailed

WV_Terminate

```
WViOsApiStatus WV_Terminate();
```

Terminate the Widevine stream control API and release all resources.

After calling this function, WV_Initialize must be called to allow other WV functions to be called again within the same application session.

Parameters

None

Returns

WViOsApiStatus_OK on success, otherwise a WViOsApiStatus error.

Events

WViOsApiEvent_Terminated

WV_SetUserData

```
WViOsApiStatus WV_SetUserData(NSString *userData);
```

Sets a new value for the WVCAUserDataKey. WV_SetUserData can be used in place of calling WV_Terminate followed by WV_Initialize to change the WVCAUserDataKey.

Parameters

[in] userData – a new value for WVCAUserDataKey

Returns

WViOsApiStatus_OK

Events

None

WV_SetCredentials

```
WViOsApiStatus WV_SetCredentials(NSDictionary *settings);
```

Change or add new optional user data values that had been passed by WV_Initialize. WV_SetCredentials can be used in place of calling a pair of methods (WV_Terminate followed by WV_Initialize) to change or add values.

Parameters

[in] settings –optData name/value pairs.

Returns

WViOsApiStatus_OK

Events

None

WV_GetDeviceId

```
NSString *WV_GetDeviceid();
```

Returns a unique device identifier. Widevine recommends using this function to obtain and set device identification as this will be different for each install of the application.

Parameters

None

Returns

Unique device identifier

Events

None

WV_RegisterAsset

```
WViOsApiStatus WV_RegisterAsset(NSString *asset);
```

Registers an asset with the Widevine library. This function does not manage the physical media file; it is the responsibility of the application to retain the actual file.

This function can be called for an already registered asset, for example, to re-license an asset with an expired license or check for license revocation.

Parameters

[in] asset – One of:

- *The relative path to the file containing a local asset.*
- *The HTTP or HTTPS URL of a remote asset.*

Returns

WViOsApiStatus_OK on success, otherwise one of the WViOsApiStatus values indicating the specific error.

Events

WViOsApiEvent_Registered

WViOsApiEvent_EMMReceived

WViOsApiEvent_EMMFailed

WV_UnregisterAsset

```
WViOsApiStatus WV_UnregisterAsset(NString *asset);
```

Unregisters an asset from the Widevine library. All information related to the file, including license, is removed from the device.

Parameters

[in] asset – One of:

- *The relative path to the file containing a local asset.*
- *The HTTP or HTTPS URL of a remote asset.*

Returns

WViOsApiStatus_OK on success, otherwise one of the WViOsApiStatus values indicating the specific error.

Events

WViOsApiEvent_EMMRemoved

WViOsApiEvent_Unregistered

WV_QueryAssetsStatus

```
WViOsApiStatus WV_QueryAssetsStatus();
```

Checks the status of all assets.

This function is used on boot to check which assets have EMMs and which assets need an EMM. Each asset calls the callback to give its status. When all assets have been reported, the status callback is called with WViOsApiEvent_EndOfList.

Parameters

none

Returns

WViOsApiStatus_OK on success, otherwise one of the WViOsApiStatus values indicating the specific error.

Events

WViOsApiEvent_QueryAsset
WViOsApiEvent_EndOfList

WV_QueryAssetStatus

```
WViOsApiStatus WV_QueryAssetStatus(NSString *asset);
```

Checks a single asset and calls the callback to obtain the asset status.

Parameters

[in] asset – One of:

- *Relative path to the file containing a local asset.*
- *HTTP or HTTPS URL of a remote asset.*

Returns

WViOsApiStatus_NotRegistered if the asset is not registered.

WViOsApiStatus_OK if the asset is registered.

Events

WViOsApiEvent_QueryAsset

WV_NowOnline

```
WViOsApiStatus WV_NowOnline();
```

Called when there is a network connection. This allows the Widevine client to retrieve EMMs for assets. The callback is called once for each asset that needs a license and refreshes each license.

Parameters

none

Returns

WViOsApiStatus_OK on success, otherwise the WViOsApiStatus value for the specific error.

Events

WViOsApiEvent_EMMReceived
WViOsApiEvent_EMMFailed

WV_Play

```
WViOsApiStatus WV_Play(NSString *asset,
                       NSMutableString *url, NSData *authentication);
```

Starts streaming the asset. The returned url contains the url that needs to be passed to the MPMoviePlayer.

Note: Only one asset may be played at a time.

Parameters

[in] asset – One of:

- *Relative path to the file containing a local asset.*
- *HTTP or HTTPS URL of a remote asset.*

[out] url – The url to pass to the MPMoviePlayer
[in] authentication – An optional encrypted block of data as described in Authentication.

Returns

WViOsApiStatus_OK on success, otherwise one of the WViOsApiStatus values indicating the specific error.

Events

WViOsApiEvent_Playing
WViOsApiEvent_PlayFailed

WV_Stop

```
WViOsApiStatus WV_Stop();
```

Stops playing or streaming the current asset.

Parameters

None

Returns

WViOsApiStatus_OK on success, otherwise one of the WViOsApiStatus values indicating the specific error.

Events

WViOsApiEvent_Stopped

WV_SelectBitRateTrack

```
WViOsApiStatus WV_SelectBitrateTrack(int trackNumber);
```

Sets the bit rate. If the specified track number is different than the currently playing track number, playback is set to the bit rate for the input track and adaptive streaming is disabled. If the specified track number is the current track number, adaptive streaming is enabled.

Parameters

[In] trackNumber – index into the bit rate list that is provided with WViOsApiEvent_Bitrates,

Returns

WViOsApiStatus_OK on success, otherwise one of the WViOsApiStatus values indicating the specific error.

Events

None

WV_SelectAudioTrack

```
WViOsApiStatus WV_SelectAudioTrack(int trackNumber);
```

Sets the audio track for the given clip. The order of the track is determined by the order the streams are listed in the SMIL file that is being read.

Parameters

[In] trackNumber – index into the audio track list that is provided with WViOsApiEvent_AudioOnlyTracks.

Returns

WViOsApiStatus_OK on success, otherwise one of the WViOsApiStatus values indicating the specific error.

Events

None

Multi-Track Audio and Subtitles

The addition of multi-track audio and subtitle support to the Widevine iOS client can be found in version 6.0.0 and above.

Playback for multi-track audio and subtitles are only supported in a Widevine multi-file package while streaming. Offline playback is not supported due to the relative path linking within the SMIL file.

See the Widevine VOD Packager API document for more details on multi-file packaging to support separate audio tracks and subtitles.

Subtitles

This feature enables the selection and playback of a subtitle XML file to accompany video playback.

The WViOsApiEvent_Subtitles event will check the SMIL file for all existing XML files and parse them out separately by assigning a unique string ID based on the name provided in the SMIL file (WVSubtitlesIdKey). This key will be used to make calls to allow switching between subtitle streams.

The following methods have been added for Subtitle support:

Method	Description
`WViOsApiEvent_Subtitles`	Lists each available Subtitle found within the provided AssetKey (i.e URL to SMIL File).
`WVMediaSourceKey`	Contains an array with each URL listed within the SMIL file. The value will be a complete URL string, which appends the root hostname with the XML file.
`WVSubtitlesIdKey`	Prints out system Language Key specified within the SMIL file (i.e “English”). Recommended Index value to be used.
`WVSubtitlesFormatKey`	Identifier value for the subtitles to list the type of format of the XML in numerical form. Standard value is 1. This should NOT be used as an index ID.

MultiTrack Audio

This feature enables the selection and playback of an audio track to accompany video playback.

The WViOsApiEvent_AudioOnlyTracks event will check the SMIL file for the “audio src” tag and parse each track to be retrieved by the WVAudioIdKey. This key will be used to identify the separate audio tracks and will be referenced when switching between audio streams.

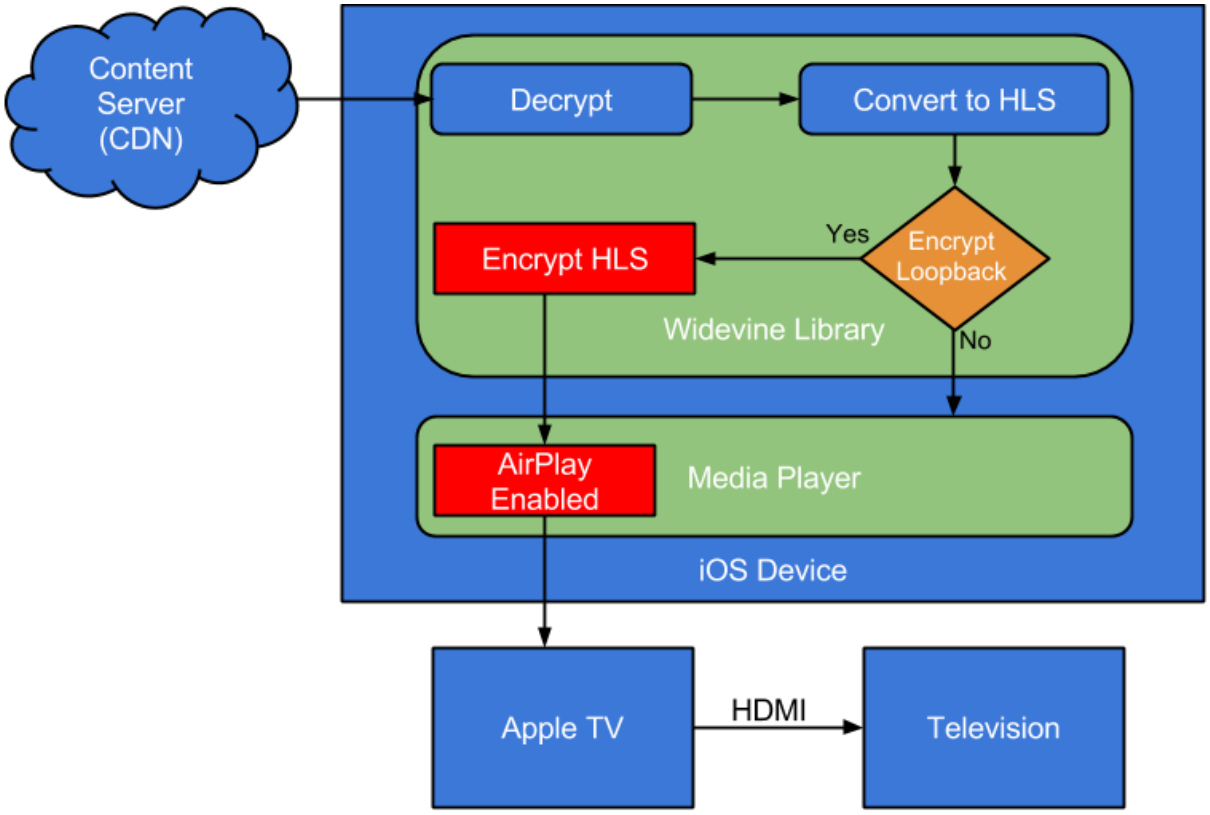
Method	Description
`WViOsApiEvent_AudioOnlyTracks`	Lists each available Audio Only track found within the provided AssetKey (i.e URL to SMIL File).
`WVAudioIdKey`	Contains an array with track’s System Language name that is listed within the SMIL file.
`WVTrackIdKey`	Contains a unique identifier to be used for switching audio. Starting value is determined by the amount of individual video bitrates. (i.e If 4 video bitrates, then first value of audio only track will be 5.)

AirPlay

The Widevine iOS client no longer denies the ability to stream content securely to an AppleTV device via the AirPlay feature. A new configuration has been added to WV_Settings that will enable an encrypted mechanism to allow AirPlay transmission.

This AirPlay setting is **disabled by default** and **must be specified via WV_Settings** (See WVUseEncryptedLoopback for more information)

It is recommended to fully disable the AirPlay feature via the user interface if AirPlay functionality is **not desired** by adding the ***allowsAirPlay property*** to your project and set the value to NO. Refer to Apple's MPMoviePlayerController Reference for further details.



Usage Guidelines

Application Development

WViOsApiEvent calls your code using a dictionary of attributes.

- The iOS requires all GUI calls to be on the main thread. Trying to change buttons or images from another thread can cause a crash.
- The Widevine callback occurs on the main thread and it has its own autorelease pool.
- The dictionary passed to the callback is autoreleased. PerformSelector does not retain it; therefore, it must be retained and released by your code.
- It is the responsibility of the application to retain, release, show and hide the MPMoviePlayer while using the library. This is handled via Apple attributes.
- Asset registration can be manually initiated by calling WV_RegisterAsset or WV_NowOnline to launch the license retrieval process.

Include WViPhoneAPI.h

```
#import "WViPhoneAPI.h"
```

Initializing

The WV_Initialize call to the Widevine iOS API is recommended to use the following settings:

- `WVDeviceIdKey`
- `WVCAUserDataKey`
- `WVPortalKey`
- `WVClientIDKey`

Upgrading from 5.0 to 6.0

Due to the introduction of Armv7s support and XML Subtitle support, the following libraries are now recommended to be included with the Demo Application.

- libc++.dylib (formerly libstdc++.dylib -- REQUIRED)
- libxml2.dylib
- libresolv.dylib
- libSystem.dylib
- libcommonCrypto.dylib
- libz.dylib

IMPORTANT NOTE: For versions prior to 6.0.0.12564, the "`WVUseJSONKey`" flag must be explicitly defined, see WV_Settings.

Bitrates

The bitrates within a WVM file or SMIL file are by default listed in order it was originally packaged (encrypted). The packaging order sequence is not necessarily in sequence (e.g. low to high bitrate).

The following Widevine methods report or set bit rates.

Method	Description
`WViOsApiEvent_Bitrates`	Lists the available bitrates within the asset and Widevine adapts to the best bit rate. The values are in bits, not bytes. Exception for 3GS, which has a 2 mpbs limit.
`WViOsApiEvent_SetCurrentBitrate`	Adjusts the bit rate, this function may be called several times after initial playback while the best bitrate is determined.
`WV_SelectBitrateTrack`	Enables the user to select a bit rate. This function locks on the specified bitrate and disables adaptive playback. Supported when using Widevine Adaption only.

Tip for Widevine Sample Demo Application:

Although UISegmentedController enables a user to select a specific bit rate, Widevine discourages exposing this much control to the user as it could adversely affect the bandwidth shifting.

Any change to a UISegmentedController, whether user selection or code change, generates an action. To avoid an endless loop, use settingBitRateButton_ as a guard.

Client Errors

EMM (License Request) Errors

A license request may fail for many reasons, including errors in CGI web interface, transport, or denial of license. For license request errors, an EMMFailed callback is generated. There are two categories of errors:

- An HTTP error below 600 or a CFStream networking error is considered a transport error and does not affect the stored license.
- An HTTP error of 600 or greater, a malformed response, or a response with result code other than 1 invalidates (revokes) any stored license.

It is the client's responsibility to call WV_QueryAsset to obtain the state of the asset. For more information regarding errors see Widevine's Status and Error Codes.

CFStream Error

Widevine callbacks will report back Apple generated, human-readable error messages, however occasionally the logging may show a CFStream Error.

A CFStream error indicates a low-level network transport error generated by Apple. For more details, see https://developer.apple.com/library/ios/navigation/index.html and check the CF (core foundation) Network documentation section for CFStream references. CF stream errors are sorted by domain: most TCP errors are in domain 4 and SSL layer errors are in domain 3.

Unsupported Usage

UDID

Apple deprecated the support of UDID based on hardware details in iOS 5.0 and subsequently Widevine no longer utilizes this mechanism. The response to this change is to use the UUID method of the NSUUID class to create a UUID and write it to the user defaults database.

Rooted OS and Simulator

The Widevine iOS client does not support jailbroken devices; methods are built into the library to prevent playback.

If you are receiving playback errors, be sure the device is not jailbroken.

The simulator is as secure as a jailbroken device and therefore is not supported.

However, the Widevine demo project comes with the WViPhoneAPI_stub.h file which will allow a project to build with the simulator, even though playback will be disabled.

References

Proxy Integration

Status and Error Codes

Full Revision History

Date	Version	Description	Author
11/18/2013	1.38	Updated AirPlay with Image	Tony Seaward
08/20/2013	1.37	Added AirPlay capability for encrypted loopback	Tony Seaward
07/25/2013	1.36	Updated Version Requirement for JSON	Tony Seaward
06/05/2013	1.35	Updated Requirement for JSON Fixed Requirement for Initialize Guideline	Tony Seaward
04/24/2013	1.34	Added JSON reference	Tony Seaward
04/04/2013	1.33	Clarified Bits vs Bytes in Event_Bitrates	Tony Seaward
04/04/2013	1.32	Included Required Libs to upgrade from 5.0 to 6.0	Tony Seaward
04/01/2013	1.31	Updated Images, Descriptions for WV_Initialize and WV_Terminate	Tony Seaward
03/31/2013	1.30	Formatting Changes	Alex Lee
03/26/2013	1.29	Restructured Document for improved reading flow	Tony Seaward
01/29/2013	1.28	Add Subtitles and MultiTrack Audio Support for 6.0.0 build.	Tony Seaward
01/23/2013	1.27	Remove Chaptering as this is no longer a supported feature.	Tony Seaward