SMARTCLIENTIOS

smartclip

Migration Guide for Version 3.0.0

Please exchange the old SmarclipSDKiOS.framework with the new SmarclipSDKiOS.xcframework.

After exchanging the old version of the SmarclipSDKiOS with version 3.0.0, you will experience some warnings and errors.
You will have to take the following steps to remove them.

Changes in the SCAVPlayerController implementation:

Changes in existing functions:

```
private func observeAVPlayer(change: [NSKeyValueChangeKey : Any] ) {

    ...
    case AVPlayer.TimeControlStatus.paused.rawValue:
        break
    default: //.playing
        // only handle this for changed states!
        if oldState != AVPlayer.TimeControlStatus.playing.rawValue  {

            eventListener?.playerEventCallback(with: EventTypePlay)

            //Initialize starting of progressBar
            if avPlayer.currentItem == currentAdvertisementItem {
                avPlayerListener?.videoStartsPlayback?()
            }
        }
    }
}

private func observePlayerItem(change: [NSKeyValueChangeKey : Any]) {

    ...

    case AVPlayerItem.Status.failed.rawValue:
        if oldState != AVPlayerItem.Status.failed.rawValue {
            if avPlayer.currentItem == currentAdvertisementItem {
                eventListener?.loadAdFailure()
                saveLastError()
                eventListener?.playerEventCallback(with: EventTypeError)
            } else if let item = avPlayer.currentItem, let error = item.error {
                avPlayerListener?.contentVideoError(error)
            }
        }
    default:
        break
    }
}

func startPlayback() {
```

```
    guard avPlayer.currentItem != nil else { return }

    if avPlayer.timeControlStatus != .playing {
        avPlayer.play()
    }
}

func pausePlayback() {
    guard avPlayer.currentItem != nil else { return }

    if avPlayer.timeControlStatus == .playing {
        avPlayer.pause()
    }
}

fileprivate func addAdItemObservers() {
    ...

    addPresentationSizeObserver()
}

fileprivate func removeAdItemObservers() {
    ...

    removePresentationSizeObserver()
}
```

Add a member variable:

```
fileprivate var lastError: SCMediaError?`
```

Add a function to save the lastError, depending on the place where the last error occured (either AVPlayer or AVPlayerItem):

```
private func saveLastError() {
    if avPlayer.status == AVPlayer.Status.failed {
        if let error = avPlayer.error {
            lastError = SCMediaError(error: error)
        }
    } else  if  avPlayer.currentItem?.status  ==  AVPlayerItem.Status.failed {
        if let error = avPlayer.currentItem?.error {
            lastError = SCMediaError(error: error)
        }
    }
}
```

Add following functions – for implementation details see SCAVPlayerController.swift (according to SCAdPlayerFacade changes):

```
func getPlayerSize() -> SCElementSize?

func getViewportSize() -> SCElementSize?

func loadAd(withCaptions loadDict: [AnyHashable : Any], disableSeeking: Bool,
completion:SCEmptyCompletionBlock?) -> Bool
```

Rename following functions:

from:

```
func setCurrentContentSource()
```

to:

```
func lockContent()
```

from:

```
func resetCurrentContentSource() -> Bool
```

to:

```
func releaseContent() -> Bool
```

Changes in function declarations (due to nullable/nonnull changes of

SCAdFacadeDelegate):

```
func getAdInfo(_ completion: @escaping SCAdInfoCompletionBlock)

func getAdError(_ completion: @escaping SCAdErrorCompletionBlock)

func onEventCallback(_ event: SCAdEvent)

func getError() -> String?

func displayClickThroughAlert(_ alert: UIAlertController)

func loadAd(withUrlString urlString: String, disableSeeking: Bool,
completion:SCEmptyCompletionBlock?) -> Bool
```

Change the implementation of the loadAd(withUrlString urlString: String, disableSeeking: Bool) function in the following way

```
func loadAd(withUrlString urlString: String, disableSeeking: Bool,
completion:SCEmptyCompletionBlock?) -> Bool {
    guard let urlStr = NSString.urlSaveHTTPString(with: urlString),
        let url = URL.init(string: urlStr) else { return false }

    lastError = nil

    ...

    DispatchQueue.main.async { [weak self] in
        self?.avPlayer.replaceCurrentItem(with: self?.currentAdvertisementItem)
        completion?()
    }
}
```

Change the implementation of the getError() -> String? function to return your added lastError variable or nil

```
func getError() -> String? {
    guard let lastErrorToReturn = lastError else { return nil }
    lastError = nil
    return lastErrorToReturn.description()
}
```

Changes in the SCOutstreamBaseViewController:

Changes in function declarations (due to nullability changes of SCAdListener):

```
func contentVideoError(_ error: Error)

func displayClickThroughAlert(_ alert: UIAlertController)
```

Add following functions:

```
override func viewDidAppear(_ animated: Bool) {
    super.viewDidAppear(animated)
    adController?.activate(true)
}

override func viewWillDisappear(_ animated: Bool) {
    super.viewWillDisappear(animated)
    adController?.activate(false)
}
```

If your app uses a UITabBar you have to be able to activate or deactivate the visibility observer when you select or deselect a tab under which your video view resides in a viewcontroller:

As an example this is shown inside the RefenceApp, which also uses a tabBar:

```
extension SCOutstreamBaseViewController: SCTabBarControllerDelegate        {
    func activate(_ isActive:Bool)  {
        adController?.activate(isActive)
    }
}
```

Changes in the SCSequencerViewController:

Changes in existing functions:

Change the implementation of `viewWillDisappear` to `viewDidDisappear`

```
override func viewDidAppear(_ animated: Bool) {
    super.viewDidAppear(animated)
    sequencer?.activate(true)
    guard sequencer?.isPaused() ?? false else { return }
    sequencer?.resumeSequencer()
}

override func viewDidDisappear(_ animated: Bool) {
    super.viewDidDisappear(animated)
    sequencer?.pause()
    sequencer?.activate(false)
}
```

Add following functions (as part of the SCAdSequencerDelegate implementation):

```
func getPlayerSize() -> SCElementSize? {
    let playerRect = view.convert(playerViewController.videoBounds, from:
playerViewController.view)
    return SCElementSize(viewMode: ViewModeNormal, boundingRect: playerRect)
}

func getViewportSize() -> SCElementSize? {
    return SCElementSize(viewMode: ViewModeNormal, boundingRect:
self.view.frame)
}
```

Hint: If you display the AVPlayerViewController in fullscreen, create SCElementSize with value `ViewModeFullscreen`.

```
func adStartsPlayback() {
    deleteProgressTimer()
    progressBar.isHidden = true
    createProgressTimer()
}
```

If your app uses a UITabBar you have to be able to activate or deactivate the visibility observer when you select or deselect a tab under which your video view resides in a viewcontroller:

As an example this is shown inside the RefenceApp, which also uses a tabBar:

```
extension SCSequencerViewController: SCTabBarControllerDelegate {
    func activate(_ isActive:Bool)  {
        sequencer?.activate(isActive)
    }
}
```