Adobe Mobile SDK for Roku Scene Graph

1. Background

Roku has introduced a new programming framework for developing applications. This new framework incorporates two new key concepts:

- Scene Graph rendering of the application screens
- XML configuration of the Scene Graph screens

Adobe mobile sdk is written in brightscript and uses a lot of components unavailable on an app running on scene graph (thread). Therefore, a roku app developer intending to use scene graph framework cannot call Adobe Mobile SDK APIs similar to traditional brightscript apps. The section below contains a bunch of important links for relevant reading on the same topic.

2. Important Links

1. Developing Scene graph apps
2. Scene graph Brightscript support
3. Scene graph Threads

3. Key Points

> *Several BrightScript functions and components cannot be used in Scene Graph component scripts. Many of the BrightScript components that cannot be used provide duplicate rendering functionality as Scene Graph nodes, and cannot be used for that reason. You should use the equivalent Scene Graph nodes instead, if available. Other BrightScript functions and components can only be used in Scene Graph applications in a **Task** node.*

> ### *Task Node Objects Ownership*
>
> *Since **Task** node objects are owned by the render thread, setting **Task** node object fields is done on the render thread, and all observer callbacks on the fields are executed in the render thread. The only case where observer callbacks are executed in a **Task** node object is if the observed field is in a node object owned by the **Task** node thread.*

4. Changes

There are no changes proposed to the SDK itself. The only extra step an application built on scenegraph needs to perform is call the tracking APIs on a background thread as opposed to the main render thread. Since this is a requirement all our roku customers would need to have, it would be really useful to have a "Task" node layer that a developer could plug in his or her sample application. This task node layer, we call **adobeMobileSceneGraphTracker.xml**, exposes one to one mapping fields for all the sdk brightscript APIs and is forever observing all these fields. We also provide withe a shim layer written in brightscript, we call **adobeMobileSceneGraphTrackerHelper.brs**, that provides a set of functions encapsulating the details of setting all the fields.

The render thread layer, for example, the video player group component, can call the functions passing in appropriate data at the right time. These functions internally are responsible for setting the right fields on the tracker, in the right order, thereby taking care of prerequisites. The field observers in the tracker are written to call the right APIs on background thread.

4.1 adobeMobileSceneGraphTracker.xml

adobeMobileSceneGraphTracker.xml is a "Task" node component for applications using scene graph. It exposes a set of "fields" with attached observer. When put in the main event loop, a render node can set these fields and the corresponding observers call adobe mobile sdk APIs.

The one to one mapping of all the fields is below:

Field	API	Type	Example	Prerequisites
trackAction	ADBMobile().trackAction(name, data)	bool	tracker.trackAction = true	stateEventName, stateEventData
trackState	ADBMobile().trackState(name, data)	bool	tracker.trackState = true	stateEventName, stateEventData
stateEventName	n/a	string	tracker.stateEventName = "test-state"	n/a

stateEventData	n/a	string	tracker.stateEventData = {'k1':'v1', 'k2':'v2'}	n/a
trackingIdentifier	ADBMobile().trackingIdentifier()	string	value = tracker.trackingIdentifier	n/a
userIdentifier	ADBMobile().userIdentifier()	string	value = tracker.userIdentifier	n/a
privacyStatus	ADBMobile().setPrivacyStatus	string	tracker.privacyStatus = 'optin'	n/a
debugLogs	ADBMobile().setDebugLogging	bool	tracker.debugLogs = true	n/a

Media

Field	API	Type	Example
mediaUpdateQoS	ADBMobile().mediaUpdateQoS(data)	bool	tracker.mediaUpdateQoS = true
mediaQoSData	n/a	assocarray	tracker.mediaQoSData = {'k1':'v1', 'k
mediaTrackEvent	ADBMobile().mediaTrackEvent(eventName,eventData,eventContextData)	bool	tracker.mediaTrackEvent = true
mediaTrackError	ADBMobile().mediaTrackError	bool	tracker.mediaTrackError = true
mediaTrackPause	ADBMobile().mediaTrackPause()	bool	tracker.mediaTrackPause = true
mediaTrackUnload	ADBMobile().mediaTrackUnload()	bool	tracker.mediaTrackUnload = true
mediaTrackComplete	ADBMobile().mediaTrackComplete()	bool	tracker.mediaTrackComplete = true
mediaTrackPlay	ADBMobile().mediaTrackPlay()	bool	tracker.mediaTrackPlay = true
mediaTrackStart	ADBMobile().mediaTrackStart()	bool	tracker.mediaTrackStart = true
mediaTrackLoad	ADBMobile().mediaTrackLoad(mediaContentData, mediaContextData)	bool	tracker.mediaTrackLoad = true
mediaEventName	n/a	string	tracker.mediaEventName = ADBMobile().MEDIA_AD_BREAK_S
mediaEventData	n/a	assocarray	currAdBreakInfo = adb_media_init_adbreakinfo("pod1" currPosition, 1) tracker.mediaEventData = currAdBr
mediaEventContextData	n/a	assocarray	tracker.mediaEventContextData = {' 'k2':'v2'}
mediaAdStart	ADBMobile().mediaTrackEvent(ADBMobile().MEDIA_AD_START, m.top.mediaAdData, m.top.mediaAdContextData)	bool	tracker.mediaAdStart = true
mediaAdData	n/a	assocarray	tracker.mediaAdData = {'k1':'v1', 'k2
mediaAdContextData	n/a	assocarray	tracker.mediaAdContextData = {'k1' 'k2':'v2'}
mediaAdComplete	ADBMobile().mediaTrackEvent(ADBMobile().MEDIA_AD_COMPLETE, m.top.mediaAdData, m.top.mediaAdContextData)	bool	tracker.mediaAdComplete = true
mediaAdBreakStart	ADBMobile().mediaTrackEvent(ADBMobile().MEDIA_AD_BREAK_START, m.top.mediaAdBreakData, m.top.mediaAdBreakContextData)	bool	tracker.mediaAdBreakStart = true
mediaAdBreakData	n/a	assocarray	currAdBreakInfo = adb_media_init_adbreakinfo("pod1" currPosition, 1) tracker.mediaAdBreakData = currAd
mediaAdBreakContextData	n/a	assocarray	tracker.mediaAdBreakContextData 'k2':'v2'}
mediaAdBreakComplete	ADBMobile().mediaTrackEvent(ADBMobile().MEDIA_AD_BREAK_COMPLETE, m.top.mediaAdBreakData, m.top.mediaAdBreakContextData)	bool	tracker.mediaAdBreakComplete = tr
mediaChapterStart	ADBMobile().mediaTrackEvent(ADBMobile().MEDIA_CHAPTER_START, m.top.mediaChapterData, m.top.mediaChapterContextData)	bool	tracker.mediaChapterStart = true
mediaChapterData	n/a	assocarray	tracker.mediaChapterData = {'k1':'v1

mediaChapterContextData	n/a	assocarray	tracker.mediaChapterContextData = 'k2':'v2'}
mediaChapterComplete	ADBMobile().mediaTrackEvent(ADBMobile().MEDIA_CHAPTER_COMPLETE, m.top.mediaChapterData, m.top.mediaChapterContextData)	bool	tracker.mediaChapterComplete = tr
mediaBufferStart	ADBMobile().mediaTrackEvent(ADBMobile().MEDIA_BUFFER_START, invalid, invalid)	bool	tracker.mediaBufferStart = true
mediaBufferComplete	ADBMobile().mediaTrackEvent(ADBMobile().MEDIA_BUFFER_COMPLETE, invalid, invalid)	bool	tracker.mediaBufferComplete = true
mediaSeekStart	ADBMobile().mediaTrackEvent(ADBMobile().MEDIA_SEEK_START, invalid, invalid)	bool	tracker.mediaSeekStart = true
mediaSeekComplete	ADBMobile().mediaTrackEvent(ADBMobile().MEDIA_SEEK_COMPLETE, invalid, invalid)	bool	tracker.mediaSeekComplete = true
mediaChapterSkip	ADBMobile().mediaTrackEvent(ADBMobile().MEDIA_CHAPTER_SKIP, invalid, invalid)	bool	tracker.mediaChapterSkip = true
mediaAdBreakSkip	ADBMobile().mediaTrackEvent(ADBMobile().MEDIA_AD_BREAK_SKIP, invalid, invalid)	bool	tracker.mediaChapterSkip = true
mediaAdSkip	ADBMobile().mediaTrackEvent(ADBMobile().MEDIA_AD_SKIP, invalid, invalid)	bool	tracker.mediaChapterSkip = true
mediaBitrateChange	ADBMobile().mediaTrackEvent(ADBMobile().MEDIA_BITRATE_CHANGE, invalid, invalid)	bool	tracker.mediaChapterSkip = true
mediaContextData	n/a	assocarray	mediaContextData = {} mediaContextData["videotype"] = "e tracker.mediaContextData = mediaContextData
mediaContentData	n/a	assocarray	mInfo = adb_media_init_mediainfo("test_me "test_media_id", 10, "vod") tracker.mediaContentData = mInfo
playhead	ADBMobile().mediaUpdatePlayhead(playhead)	float	tracker.playhead = 10.1
mediaErrorMessage	n/a	string	tracker.mediaErrorMessage = 'very
mediaErrorCode	n/a	string	tracker.mediaErrorCode = '100'

Audience

Field	API	Type	Example	Prerequisites
audienceSetDpidAndDpuuid	ADBMobile().audienceSetDpidAndDpuuid(dpid,dpuuid)	bool	tracker.audienceSetDpidAndDpuuid = true	audienceDpid, audienceDpuuid
audienceSubmitSignal	ADBMobile().audienceSubmitSignal(traits)	bool	tracker.audienceSubmitSignal = true	traits
audienceVisitorProfile	ADBMobile().audienceVisitorProfile()	assocarray	value = tracker.audienceVisitorProfile	n/a
audienceDpid	ADBMobile().audienceDpid()	string	value = tracker.audienceDpid	n/a
audienceDpuuid	ADBMobile().audienceDpuuid()	string	value = tracker.audienceDpuuid	n/a
traits	n/a	assocarray	tracker.traits = data	n/a

Visitor

Field	API	Type	Example	Prerequisites
visitorSyncIdentifiers	ADBMobile().visitorSyncIdentifiers(visitorIdentifiers)	bool	tracker.visitorSyncIdentifiers = true	visitorIdentifiers
visitorMarketingCloudID	ADBMobile().visitorMarketingCloudID()	string	value = tracker.visitorMarketingCloudID	n/a
visitorIdentifiers	n/a	assocarray	tracker.visitorIdentifiers = {'k1':'v1', 'k2':'v2'}	n/a

4.2 adobeMobileSceneGraphTrackerHelper.brs

adobeMobileSceneGraphTrackerHelper.brs is a brightscript shim layer that exposes intuitive functions that a developer can call instead of setting individual fields on "adobeMobileSceneGraphTracker" described above. The helper functions can deal with the fields and set them in the right order as expected.

A developer can create an instance of adobeMobileSceneGraphTrackerHelper by passing it an instance of adobeMobileSceneGraphTracker. Calling functions on the helper then implements setting of fields on the instance that's passed along.

Details of all the functions signatures, as well as which fields they set on the tracker instance, are below:

Constructor

Function	Parameters	Returns	Tracker Fields Set
initMediaTracker	mobileTracker as Object	void	n/a

Analytics

Function	Parameters	Returns	Tracker Fields Set
trackAction	actionName as String actionData as String	void	m.tracker.stateEventName = actionName m.tracker.stateEventData = actionData m.tracker.trackAction = true
trackState	stateName as String stateData as String	void	m.tracker.stateName = actionName m.tracker.stateData = actionData m.tracker.trackState = true
debugLogging	flag as Boolean	void	m.tracker.debugLogs = flag
trackingIdentifier	void	string	m.tracker.trackingIdentifier
userIdentifier	void	string	m.tracker.userIdentifier
setPrivacyStatus	status as String	void	m.tracker.privacyStatus = status
getPrivacyStatus	void	string	return m.tracker.privacyStatus

Media

Function	Parameters	Returns	Tracker Fields Set
trackMediaQos	qosData	void	m.tracker.mediaQoSData = qosData m.tracker.mediaUpdateQoS = true
trackMediaLoad	contentData, contextData As Object	void	m.tracker.mediaContentData = contentData m.tracker.mediaContextData = contextData m.tracker.mediaTrackLoad = true
trackMediaStart	none	void	m.tracker.mediaTrackStart = true
trackMediaPlay	none	void	m.tracker.mediaTrackPlay = true
enableDebugLogging	none	none	m.tracker.debugLogs = true
runMediaTracker	none	none	m.tracker.control = "RUN"
updatePlayhead	position as float	none	m.tracker.playhead = position
trackMediaEvent	mediaEventName as String, mediaEventInfo as Object, mediaEventContextData as Object	none	m.tracker.mediaEventName = mediaEventName m.tracker.mediaEventData = mediaEventInfo m.tracker.mediaEventContextData = mediaEventContextData m.tracker.mediaTrackEvent = true

trackError	errorMsg as String, errorCode as String	none	m.tracker.errorMessage = errorMsg m.tracker.errorCode = errorCode m.tracker.mediaTrackError = true
trackMediaComplete	none	none	m.tracker.mediaTrackComplete = true
trackMediaUnload	none	none	m.tracker.mediaTrackUnload = true
trackMediaPause	none	none	m.tracker.mediaTrackPause = true

Audience Manager

Function	Parameters	Return	Tracker Fields set
audienceSetDpidAndDpuuid	audienceDpid as String, audienceDpuuid as String	void	m.tracker.audienceDpid = audienceDpid m.tracker.audienceDpuuid = audienceDpuuid m.tracker.audienceSetDpidAndDpuuid = true
audienceSubmitSignal	traits as Object	void	m.tracker.traits = traits m.tracker.audienceSubmitSignal = true
audienceDpid	void	string	return m.tracker.audienceDpid
audienceDpuuid	void	string	return m.audienceDpuuid

Visitor service

Function	Parameters	Return	Tracker Fields Set
visitorSyncIdentifiers	visitorIdentifiers as Object	void	m.tracker.visitorIdentifiers = visitorIdentifiers m.tracker.visitorSyncIdentifiers = true
visitorMarketingCloudID	void	string	return m.tracker.visitorMarketingCloudID

4.3 Sample Use

```
<component name = "VideoExample" extends = "Group" initialFocus =
"videoPlayer" >

  <script type="text/brightscript" uri="pkg:/source/adbmobile.brs" />
  <script type="text/brightscript"
uri="pkg:/components/adobeMobileSceneGraphTrackerHelper.brs" />

  <script type = "text/brightscript" >

    <![CDATA[

    sub init()
      m.top.streamStarted = "false"

      videocontent = createObject("RoSGNode", "ContentNode")
      videocontent.title = "Example Video"
      videocontent.streamformat = "mp4"
      videocontent.url = INSERT_CONTENT_URL_HERE
      m.top.videoPlayer = m.top.findNode("videoPlayer")
      m.top.videoPlayer.visible = true
      m.top.videoPlayer.setFocus(true)
      m.top.videoPlayer.content = videocontent
```

```
      m.top.adobeTrackerTask =
createObject("RoSGNode","AdobeMobileSceneGraphTracker")

      m.top.videoPlayer.observeField("state", "OnVideoPlayerStateChange")
      m.top.videoPlayer.control = "play"

      initMediaTracker(m.top.adobeTrackerTask)
      runMediaTracker()
    end sub

 Sub OnVideoPlayerStateChange()
      m.videoPlayer = m.top.videoPlayer

      if m.videoPlayer <> invalid
        if m.videoPlayer.state = "error"
          trackError(m.videoPlayer.errorMsg, m.videoPlayer.errorCode)
          m.videoPlayer.visible = false

        else if m.videoPlayer.state = "playing"

          if m.top.streamStarted = "false"
            m.top.streamStarted = "true"

            mInfo = adb_media_init_mediainfo("test_media_name",
"test_media_id", 10, "vod")
            mediaContextData = {}
            mediaContextData["videotype"] = "episode"

            trackMediaLoad(mInfo, mediaContextData)
            trackMediaStart()
          endif

            trackMediaPlay()
```

. . . . . .
. . . . . .
. . . . . .