Adobe® Marketing Cloud

Roku SDK 2.x for Marketing Cloud Solutions

Table of Contents

ROKU SDK 2.X FOR MARKETING CLOUD SOLUTIONS **3**
Adobe Mobile Services *3*
GETTING STARTED 4
ANALYTICS 5
Track App States *5*
Track App Actions *5*
MARKETING CLOUD 6
Marketing Cloud Visitor ID Configuration *6*
Marketing Cloud Visitor ID Service Methods *6*
Postbacks *6*
AUDIENCE MANAGER 7
Audience Manager Methods *7*
VIDEO ANALYTICS VIA MEDIA HEARTBEATS 8
Integrating media heartbeats APIs *8*
OPT-OUT AND PRIVACY SETTINGS 16
DEBUG LOGGING 16
USING BLOODHOUND TO TEST ROKU APPLICATIONS 17
CONTACT AND LEGAL INFORMATION 18
Help & Technical Support *18*
Service, Capability & Billing *18*
Feedback *18*
Legal *18*

Roku SDK 2.x for Marketing Cloud Solutions

Roku SDK 2.x for Marketing Cloud Solutions lets you measure Roku applications written in BrightScript, leverage and collect audience data through audience management, and measure video engagement through Video heartbeats.

Adobe Mobile Services

Adobe Mobile services provides a new UI that brings together mobile marketing capabilities for mobile applications from across the Adobe Marketing Cloud. Initially, the Mobile service provides seamless integration of app analytics and targeting capabilities from the Adobe Analytics and Adobe Target solutions. Learn more at **Adobe Mobile Services documentation.**

Getting Started

Information to help you get starting with the Roku SDK for Marketing Cloud Solutions. This section assumes that you have configured a report suite through Adobe Mobile Services to collect app data.

Adobe Mobile services is the primary reporting interface for mobile app analytics and targeting. After you complete the configuration steps you can download a configuration file that is pre-configured with your data collection server, report suite, and many other settings.

Get the SDK

After unzipping the AdobeMobileLibrary-2.*-Roku.zip download file, you'll have the following software components:

- *adbmobile.brs*: Library file to be included in your Roku app source folder.
- *ADBMobileConfig.json*: SDK configuration file customized for your app.

Add the library file and json config file to your project source.

Analytics

Information to help you use the Roku SDK with Adobe Analytics.

Track App States

States are the different screens or views in your application.

Each time a new state is displayed in your application (for example, when a user navigates from the home page to the video details screen), you should send a *trackState* call. In Roku, *trackState* is typically called each time a new screen is loaded.

```
ADBMobile().trackState("State Name", {})
```

The "State Name" is reported in the View State variable in Adobe Mobile services, and a view is recorded for each *trackState* call. In other Analytics interfaces, View State is reported as Page Name and state views is reported as page views.

In addition to the "State Name", you can send additional context data with each track action call:

```
dictionary = { }
dictionary["myapp.login.LoginStatus"] = "logged in"
ADBMobile().trackState("Home Screen", dictionary)
```

Context data values must be mapped to custom variables in Adobe Mobile services.

States are typically viewed using a pathing report so you can see how users navigate your app, and which states are viewed most.

Track App Actions

Actions are the events that occur in your app that you want to measure.

Each action has one or more corresponding metrics that are incremented each time the event occurs. For example, you might send a *trackAction* call for each new subscription, each time a content is rated, or each time a level is completed. Actions are not tracked automatically, you must call *trackAction* when an event you want to track occurs, and then map the action to a custom event.

```
ADBMobile().trackAction("myapp.ActionName", {})
```

In addition to the action name, you can send additional context data with each track action call:

```
dictionary = {}
dictionary["myapp.social.SocialSource"] = "Twitter"
ADBMobile().trackAction(""myapp.SocialShare", dictionary)
```

Marketing Cloud

Information to help you use the Roku SDK with the Adobe Marketing Cloud.

Marketing Cloud Visitor ID Configuration

The Marketing Cloud visitor ID service provides a universal visitor ID across Marketing Cloud solutions. The visitor ID service is required by Video heartbeat and future Marketing Cloud integrations.

Verify that ADBMobileConfig.json contains the marketingCloudorg:

```
"marketingCloud" : {
 "org": "YOUR-MCORG-ID"
 }
```

Note: Marketing Cloud organization IDs uniquely identify each client company in the Adobe Marketing Cloud, and are similar to the following value: 016D5C175213CCA80A490D05@AdobeOrg. Be sure to include @AdobeOrg.

After configuration, a Marketing Cloud visitor ID is generated and is included on all hits. Other visitor IDs (custom and automatically-generated) continue to be sent with each hit.

Marketing Cloud Visitor ID Service Methods

Marketing Cloud visitor ID methods are prefixed with "visitor."

Method	Description
visitorMarketingCloudID	Retrieves the Marketing Cloud visitor ID from the visitor ID service. *ADBMobile().visitorMarketingCloudID()*
visitorSyncIdentifiers	Along with the Marketing Cloud visitor ID, you can set additional customer IDs to associate with each visitor. The Visitor API accepts multiple Customer IDs for the same visitor, along with a customer type identifier to separate the scope of the different customer IDs. This method corresponds to setCustomerIDs in the JavaScript library. Ex: *identifiers = {}* *identifiers["idType"] = "idValue"* *ADBMobile().visitorSyncIdentifiers(identifiers)*

Postbacks

Please refer Postbacks configuration online documentation source here:

https://marketing.adobe.com/resources/help/en_US/mobile/signals_.html

Audience Manager

Information to help you send signals and retrieve visitor segments from audience manager.

Audience Manager Methods

Send signals and retrieve visitor segments from audience management.

Method	**Description**
audienceVisitorProfile	Returns the visitor profile that was most recently obtained. Returns an empty object if no signal has been submitted yet. *ADBMobile().audienceVisitorProfile()*
audienceDpid	Returns the current DPID. *ADBMobile().audienceDpid()*
audienceDpuuid	Returns the current DPUUID. *ADBMobile().audienceDpuuid()*
audienceSetDpidAndDpuuid	Sets the DPID and DPUUID. If DPID and DPUUID are set, they will be sent with each signal. *ADBMobile().audienceSetDpidAndDpuuid("myDpid", "myDpuuid")*
audienceSubmitSignal	Sends audience management a signal with traits *ADBMobile().audienceSubmitSignal()*

Video Analytics via Media heartbeats

Information to instrument a media (video/audio) player with the media module of Adobe Mobile SDK.

The APIs belonging to media heartbeats enable real-time dashboards and other media reporting capabilities. This guide is intended for a media integration engineer who has an understanding of the APIs and workflow of the media player being instrumented. Implementing these APIs requires that your media player provide the following:

- An API to subscribe to player events. The media heartbeat requires that you call a set of simple APIs when events occur in your player.
- An API or class that provides player information, such as media name and play-head position.

Integrating media heartbeats APIs

Integrating media analytics real-time media tracking into a media player requires including Adobe Mobile SDK, instantiating and configuring the media heartbeats instance, listening to media player events and using appropriate media heartbeats APIs in your project. Therefore the steps for a developer integrating the media heartbeats APIs are as follows:

Initial Setup

1. Acquire the required Adobe Mobile SDK and add it into your project: The SDK contains the media heartbeats module.
2. Provide the configuration for heartbeats through ADBMobileConfig.json. The new media module, self configures itself through ADBMobileConfig json.

Approaches to tracking Media during a Playback Session

There are two ways you can track your media content through heartbeats

1. Using *ADBVideoPlayer*: The media heartbeats module provides an extension of the standard native player that takes care of binding the native player events to media module tracking APIs. If the application is built using this standard media player (for example: roVideoScreen on Roku) , it is highly recommended that the developer let the provided extension handle all incoming events from the player by passing all incoming events to the extension as is. The extension takes care of calling relevant media heartbeats APIs based on the event type. This saves the developers a lot of effort in handling individual player events themselves. This way of integration is also demonstrated in the demo player provided with the SDK.
2. Using Tracking APIs: This way of integration involves handling player events and calling relevant media heartbeats APIs from the application. The APIs documentation and this guide should be used to learn the details on all the media heartbeats APIs and their respective life cycle.

Configure Media heartbeats

Once the application has acquired the Adobe Mobile SDK, the first step is to configure the media heartbeats.

The JSON used to configure Adobe Mobile has an exclusive key for media heartbeats with the name "mediaHeartbeat". This is where the configuration parameters for the media heartbeats belong. An example below demonstrates it. Also, a sample ADBMobileConfig JSON file is provided with the package, settings for which must be obtained from an Adobe representative.

ADBMobileConfig.json

```
{
  "version":"1.0",
  "analytics":{
    "rsids":"",
    "server":"",
    "charset":"UTF-8",
    "ssl":false,
    "offlineEnabled":false,
    "lifecycleTimeout":30,
    "batchLimit":50,
    "privacyDefault":"optedin",
    "poi":[

    ]
  },
  "marketingCloud":{
    "org":""
  },
  "target":{
    "clientCode":"",
    "timeout":5
  },
  "audienceManager":{
    "server":""
  },
  "acquisition":{
    "server":"example.com",
    "appid":"sample-app-id"
  },
  "mediaHeartbeat":{
    "server":"example.com",
    "publisher":"sample-publisher",
    "channel":"sample-channel",
    "ssl":false,
    "ovp":"sample-ovp",
    "sdkVersion":"sample-sdk",
    "playerName":"roku"
  }
}
```

Config Parameter	Description
server	String representing the URL of the tracking endpoint on the backend side.
publisher	String representing the content publisher unique identifier.
channel	String representing the name of the content distribution channel.
ssl	Boolean representing whether ssl should be used for tracking calls
ovp	String representing the name of the video player provider
sdkVersion	String representing the current version of the app/sdk.
playerName	String representing the name of the player.

If *mediaHeartbeat* is incorrectly configured, media module (VHL) will go into an error state and will not send any tracking calls.

Media heartbeat methods

Track media using the following methods

Method	Description
processMessages	This method processes queue to send out analytics hits except for Media . Must be called in every screen event loop where analytics hits are being sent *Ex:* *while true* *msg = wait(100, screen.GetMessagePort())* *' Call this in every screen event loop* *ADBMobile().processMessages()* *.....*
processMediaMessages	This method processes queue to send out Media tracking hits . Must be called in every video screen event loop where Media heartbeat hits are being sent *Ex:* *ADBMobile().processMediaMessages()*
mediaTrackLoad	Media playback tracking method to track Media Load, and set the current session active Ex: *‘ Create a media info object* *mediaInfo = adb_media_init_mediainfo()* *mediaInfo.id = "sample-media-id"* *mediaInfo.playhead = "0"* *mediaInfo.length = "600"* *‘ Create context data if any* *mediaContextData = {}* *mediaContextData["cmk1"] = "cmv1"* *mediaContextData["cmk2"] = "cmv2"* *ADBMobile().mediaTrackLoad(mediaInfo,mediaContextData)*
mediaTrackStart	Media playback tracking method to track Session Start Ex: *ADBMobile().mediaTrackStart()*
mediaTrackUnload	Media playback tracking method to track Media Unload and deactivate current session Ex: *ADBMobile().mediaTrackUnload()*
mediaTrackPlay	Media playback tracking method to track Media Play Ex: *ADBMobile().mediaTrackPlay()*
mediaTrackPause	Media playback tracking method to track Media Pause Ex: *ADBMobile().mediaTrackPause()*
mediaTrackComplete	Media playback tracking method to track Media Complete Ex: *ADBMobile().mediaTrackComplete()*
mediaTrackError	Error tracking method to track Player Error Ex: *ADBMobile().mediaTrackError(msg.GetMessage(),*

	ADBMobile().ERROR_SOURCE_PLAYER)
mediaTrackEvent	Media tracking method to track events that do not belong to media lifecycle and are optional. Example: AD_START/AD_COMPLETE, CHAPTER_START/CHAPTER_COMPLETE; Refer Events section for detailed list of events. This method takes three arguments: *event constant, event info, and context data (send empty object if there is no context data)*

mediaUpdatePlayhead	Method to report playhead position updates. This method must be called from application to report every update on playhead position
	Ex:
	if (mInfo.streamType = ADBMobile().MEDIA_STREAM_TYPE_LIVE)
	ADBMobile().mediaUpdatePlayhead(-1)
	else
	ADBMobile().mediaUpdatePlayhead(msg.GetIndex())
	endif

mediaUpdateQoS	Method to report QoS metrics updates. This method must be called from application to report every update on QoS metrics Ex: *qosInfo = adb_media_init_qosinfo()* *qosInfo.droppedFrames = 1* *qosInfo.startupTime = 2* *qosInfo.fps = 0* *qosInfo.bitrate = 200000* *ADBMobile().mediaUpdateQoS(qosInfo)*

In addition to the above methods, the media module also provides constants to track media events

Constant	**Description**
MEDIA_BUFFER_START	EventType for Buffer Start
MEDIA_BUFFER_COMPLETE	EventType for Buffer Complete
MEDIA_SEEK_START	EventType for Seek Start
MEDIA_SEEK_COMPLETE	EventType for Seek Complete
MEDIA_BITRATE_CHANGE	EventType for Bitrate change
MEDIA_CHAPTER_START	EventType for Chapter Start
MEDIA_CHAPTER_COMPLETE	EventType for Chapter Complete
MEDIA_CHAPTER_SKIP	EventType for Chapter skip
MEDIA_AD_BREAK_START	EventType for AdBreak Start
MEDIA_AD_BREAK_COMPLETE	EventType for AdBreak Complete
MEDIA_AD_BREAK_SKIP	EventType for AdBreak Skip
MEDIA_AD_START	EventType for Ad Start
MEDIA_AD_COMPLETE	EventType for Ad Complete

MEDIA_AD_SKIP	EventType for Ad Skip
MEDIA_STREAM_TYPE_LIVE	Constant for Stream Type LIVE
MEDIA_STREAM_TYPE_VOD	Constant for Stream Type VOD
ERROR_SOURCE_PLAYER	Constant for Error source being Player
MEDIA_STANDARD_VIDEO_METADATA	Constant to set Video Metadata on MediaInfo object in trackLoad API
MEDIA_STANDARD_AD_METADATA	Constant to set Ad Metadata on EventData object in trackEvent API for Ad start

Standard Metadata

Standard video and ad metadata can be set on media and ad info objects respectively. Using the constants keys for video/ad metadata set the dictionary containing standard metadata on info object before calling the track APIs. Refer the tables below for the entire list of standard metadata constants, followed by sample implementation.

Standard Video Metadata Constants

Metadata Name	Context Data Key	Constant Name
Show	a.media.show	MEDIA_VideoMetadataKeySHOW
Season	a.media.season	MEDIA_VideoMetadataKeySEASON
Episode	a.media.episode	MEDIA_VideoMetadataKeyEPISODE
Asset	a.media.asset	MEDIA_VideoMetadataKeyASSET_ID
Genre	a.media.genre	MEDIA_VideoMetadataKeyGENRE
First Air Date	a.media.airDate	MEDIA_VideoMetadataKeyFIRST_AIR_DATE
First Digital Air Date	a.media.digitalDate	MEDIA_VideoMetadataKeyFIRST_DIGITAL_DATE
Rating	a.media.rating	MEDIA_VideoMetadataKeyRATING
Originator	a.media.originator	MEDIA_VideoMetadataKeyORIGINATOR
Network	a.media.network	MEDIA_VideoMetadataKeyNETWORK
Show Type	a.media.type	MEDIA_VideoMetadataKeySHOW_TYPE
Ad Load	a.media.adLoad	MEDIA_VideoMetadataKeyAD_LOAD
MVPD	a.media.pass.mvpd	MEDIA_VideoMetadataKeyMVPD
Authorized	a.media.pass.auth	MEDIA_VideoMetadataKeyAUTHORIZED

Day Part	a.media.dayPart	MEDIA_VideoMetadataKeyDAY_PART
Feed	a.media.feed	MEDIA_VideoMetadataKeyFEED
Stream Format	a.media.format	MEDIA_VideoMetadataKeySTREAM_FORMAT

Standard Ad Metadata Constants

Metadata Name	Context Data Key	Constant Name
Advertiser	a.media.ad.advertiser	MEDIA_AdMetadataKeyADVERTISER
Campaign ID	a.media.ad.campaign	MEDIA_AdMetadataKeyCAMPAIGN_ID
Creative ID	a.media.ad.creative	MEDIA_AdMetadataKeyCREATIVE_ID
Placement ID	a.media.ad.placement	MEDIA_AdMetadataKeyPLACEMENT_ID
Site ID	a.media.ad.site	MEDIA_AdMetadataKeySITE_ID
Creative URL	a.media.ad.creativeURL	MEDIA_AdMetadataKeyCREATIVE_URL

Sample Implementation for Standard Metadata

```
// setting Standard Video Metadata as context data on trackLoad API
  mediaContextData = { }
  mediaContextData["videotype"] = "episode"

  standaradMetadata = {}
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeySHOW] = "sample show"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeySEASON] = "sample season"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeyEPISODE] = "sample episode"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeyTMS_ID] = "sample tms_id"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeyGENRE] = "sample genre"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeyFIRST_AIR_DATE] = "sample first_air_date"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeyFIRST_DIGITAL_DATE] = "sample first_digital_date"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeyRATING] = "sample rating"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeyORIGINATOR] = "sample originator"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeyNETWORK] = "sample network"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeySHOW_TYPE] = "sample show type"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeyAD_LOAD] = "sample ad load"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeyMVPD] = "sample mvpd"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeyAUTHORIZED] = "sample authorized"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeyDAY_PART] = "sample day_part"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeyFEED] = "sample feed"
  standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeySTREAM_FORMAT] = "sample format"

  mediaInfo = adb_media_init_mediainfo(content.name, content.id, content.length, content.streamType)
  mediaInfo[ADBMobile().MEDIA_STANDARD_VIDEO_METADATA] = standaradMetadata
  ADBMobile().mediaTrackLoad(mediaInfo, mediaContextData)

// setting Standard Ad Metadata as context data on ad start event
```

```
standardAdMetadata = {}
standardAdMetadata[ADBMobile().MEDIA_AdMetadataKeyCAMPAIGN_ID] = "sample campaign"
standardAdMetadata[ADBMobile().MEDIA_AdMetadataKeyADVERTISER] = "sample advertiser"
standardAdMetadata[ADBMobile().MEDIA_AdMetadataKeyCREATIVE_ID] = "sample creativeid"
standardAdMetadata[ADBMobile().MEDIA_AdMetadataKeyPLACEMENT_ID] = "sample placement id"
standardAdMetadata[ADBMobile().MEDIA_AdMetadataKeySITE_ID] = "sample site id"
standardAdMetadata[ADBMobile().MEDIA_AdMetadataKeyCREATIVE_URL] = "sample creative url"

adInfo = adb_media_init_adinfo(ad.title, ad.title, ad.position, ad.duration)
adInfo[ADBMobile().MEDIA_STANDARD_AD_METADATA] = standardAdMetadata
ADBMobile().mediaTrackEvent(ADBMobile().MEDIA_AD_START, adInfo, ad.contextData)
```

There is also convenience methods as described below for creating various info objects sent through the media heartbeat API methods. Please refer to the table below

Method	**Description**
adb_media_init_mediainfo	This method returns an initialized Media Information object *Function adb_media_init_mediainfo(name As String, id As String, length As Double, streamType As String) As Object*
adb_media_init_adinfo	This method returns initialized Ad Information object *Function adb_media_init_adinfo(name As String, id As String, position As Double, length As Double) As Object*
adb_media_init_chapterinfo	This method returns initialized Chapter Information object *Function adb_media_init_chapterinfo(name As String, position As Double, length As Double, startTime As Double) As Object*
adb_media_init_adbreakinfo	This method returns initialized AdBreak Information object *Function adb_media_init_adbreakinfo(name As String, startTime as Double, position as Double) As Object*
adb_media_init_qosinfo	This method returns initialized QoS Information object *Function adb_media_init_qosinfo(bitrate As Double, startupTime as Double, fps as Double, droppedFrames as Double) As Object*

Once the application developer is familiar with all the above APIs, there are two ways of integrating media heartbeats with the application media player: Using *ADBVideoPlayer* and calling raw tracking APIs directly.

Using ADBVideoPlayer

The media heartbeats module provides an extension of the standard native player that takes care of binding the native player events to media module tracking APIs. If the application is built using this standard media player (for example: roVideoScreen on Roku) , it is highly recommended that the developer let the provided extension handle all incoming events from the player by passing all incoming events to the extension as is. The extension takes care of calling relevant media heartbeats APIs based on the event type. This saves the developers a lot of effort in handling individual player events themselves. This way of integration is also demonstrated in the demo player provided with the SDK.

Any custom media events like AD_START, CHAPTER_START etc. would still be handled separately and tracked explicitly by the application using *ADBMobile().mediaTrackEvent()* APIs (see the second method below). The code snippet below shows how the application can listen to roVideoScreen events and pass the message to ADBVideoPlayer. If you intend to use standard metadata via ADBVideoPlayer, you need to create a dictionary of standard metadata and set it on content. Following code snippet demonstrate the usage:

Using Media Tracking APIs

App developers can use the media tracking APIs described above directly to track media life cycle and ad/chapter events. Please refer to the sample app for example implementation on implementing these APIs.

ADBVideoPlayer Sample

```
Function showVideoContent(content As Object)

    port = CreateObject("roMessagePort")
    screen = CreateObject("roVideoScreen")
    screen.SetMessagePort(port)

    screen.SetPositionNotificationPeriod(1)
    screen.SetContent(content)
    screen.Show()

    ''' Send the content object. This object must specify the 'live' property if the content is a live stream

    videoContextData = { }
    videoContextData ["videotype"] = "episode"

    standaradMetadata = {}
    standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeySHOW] = "sample show"
    standaradMetadata[ADBMobile().MEDIA_VideoMetadataKeySEASON] = "sample season"

    content[ADBMobile().MEDIA_STANDARD_VIDEO_METADATA] = standaradMetadata
    ADBVideoPlayer().setContent(content, videoContextData)

    while true
        msg = wait(100, port)

        ''' Let ADBVideoPlayer handle the message first
        ADBVideoPlayer().handleMessage(msg)

        ''' Run the process messages handler for ADB Media
        ADBMobile().processMediaMessages()

        if type(msg) = "roVideoScreenEvent" then
            if msg.isScreenClosed()
                exit while
            elseif msg.isRequestFailed()
            elseif msg.isStatusMessage()
            elseif msg.isButtonPressed()
            elseif msg.isPlaybackPosition() then
            end if
        else
            'print "Unexpected message class: "; type(msg)
        end if
    end while
End Function
```

Opt-Out and Privacy Settings

You can control whether or not Analytics data is sent on a specific device using the following settings:

- *privacyDefault* setting in ADBMobile JSON Config. This controls the initial setting that persists until it is changed in code.
- *ADBMobile().setPrivacyStatus()* method. After the privacy setting is changed using this method, the change is permanent until it is changed again using this method, or the app is completely uninstalled and re-installed.

Note: Media heartbeat tracking calls are also disabled if the privacy status is set to opt-out

Set this to ADBMobile().PRIVACY_STATUS_OPT_IN/ADBMobile().PRIVACY_STATUS_OPT_OUT if user wants to opt-in/opt-out
ADBMobile().setPrivacyStatus(ADBMobile().PRIVACY_STATUS_OPT_OUT)

This method will return the current value for privacy status constant (ADBMobile().PRIVACY_STATUS_OPT_IN or ADBMobile().PRIVACY_STATUS_OPT_OUT)
ADBMobile().getPrivacyStatus()

Debug Logging

ADBMobile library provides debug logging through the *setDebugLogging* method. Debug logging should be set to false for all the production apps.

ADBMobile().setDebugLogging(true)

Using Bloodhound to Test Roku Applications

During application development, Bloodhound lets you view server calls locally, and optionally forward the data to Adobe collection servers.

Bloodhound can be downloaded from any app configuration page in Adobe Mobile services.

Bloodhound 3 Beta for Mac documentation

Bloodhound 2 for Windows documentation

Contact and Legal Information

Information to help you contact Adobe and to understand the legal issues concerning your use of this product and documentation.

Help & Technical Support

The Adobe Marketing Cloud Customer Care team is here to assist you and provides a number of mechanisms by which they can be engaged:

• Check the Marketing Cloud help pages for advice, tips, and FAQs
• Ask us a quick question on Twitter @AdobeMktgCare
• Log an incident in our customer portal
• Contact the Customer Care team directly
• Check availability and status of Marketing Cloud Solutions

Service, Capability & Billing

Dependent on your solution configuration, some options described in this documentation might not be available to you. As each account is unique, please refer to your contract for pricing, due dates, terms, and conditions. If you would like to add to or otherwise change your service level, or if you have questions regarding your current service, please contact your Account Manager.

Feedback

We welcome any suggestions or feedback regarding this solution. Enhancement ideas and suggestions for the Analytics suite can be added to our Customer Idea Exchange.

Legal

© 2015 Adobe Systems Incorporated. All Rights Reserved.
Published by Adobe Systems Incorporated.
Terms of Use | Privacy Center

Adobe and the Adobe logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. A trademark symbol (®, ™, etc.) denotes an Adobe trademark.
All third-party trademarks are the property of their respective owners. Updated Information/Additional Third Party Code. Information available at http://www.adobe.com/go/thirdparty.