



Streaming Player Component Reference

Roku Streaming Player Version 4.1

12/21/2011

Copyright (c) 2009, 2010, 2011 Roku Inc. All rights reserved. Use of this Roku SDK Documentation is limited and expressly conditioned on consent to and compliance with the terms and conditions of the Roku Channel Developer Agreement.

http://www.roku.com/Libraries/Legal/Roku_Channel_Developer_Agreement.sflb.ashx

Table of Contents

| | |
|---|------------|
| 1.0 Introduction to the Roku Component Reference | 4 |
| 2.0 BrightScript Components | 4 |
| 2.1 Introduction to BrightScript Components | 4 |
| 2.2 BrightScript Component Interfaces and Methods | 4 |
| 2.3 Classes | 5 |
| 2.4 Event Loops | 5 |
| 2.5 Entry Points | 7 |
| 3.0 Roku Streaming Player Object Library | 9 |
| 3.1 Roku Streaming Player Path Names | 10 |
| 3.2 Application Directory and Manifest Data | 11 |
| 3.3 Content Meta-Data | 14 |
| 4.0 Original Platform Components (Since v2.4) | 25 |
| 4.1 General Attributes | 25 |
| 4.2 roAppManager | 26 |
| 4.3 roPosterScreen | 35 |
| 4.4 roSpringboardScreen | 43 |
| 4.5 roVideoScreen | 51 |
| 4.6 roSlideShow | 61 |
| 4.7 roSearchScreen | 64 |
| 4.8 roSearchHistory | 69 |
| 4.9 roParagraphScreen | 69 |
| 4.10 roMessageDialog | 73 |
| 4.11 roOneLineDialog | 77 |
| 4.12 roCodeRegistrationScreen | 78 |
| 4.13 roKeyboardScreen | 82 |
| 4.14 roPinEntryDialog | 85 |
| 4.15 roAudioPlayer | 86 |
| 4.16 roMessagePort | 90 |
| 4.17 roDeviceInfo | 90 |
| 4.18 roDateTime | 92 |
| 4.19 roEVPCipher | 93 |
| 4.20 roEVPDigest | 94 |
| 4.21 roHMAC | 95 |
| 4.22 roTimespan | 96 |
| 4.23 roRegistry | 97 |
| 4.24 roRegistrySection | 98 |
| 4.25 roUrlTransfer | 99 |
| 5.0 Components First Introduced in Firmware v2.6 | 106 |
| 5.1 rolmageCanvas | 106 |
| 5.2 roFontRegistry | 109 |
| 5.3 roFontMetrics | 110 |
| 5.4 roSystemLog | 111 |
| 5.5 roRegex | 114 |
| 5.6 roPath | 117 |
| 5.7 roFileSystem | 117 |
| 5.8 roVideoPlayer | 119 |

| | |
|---|------------|
| 6.0 Components First Introduced in Firmware v2.7 | 124 |
| 6.1 roGridScreen | 124 |
| 7.0 Components First Introduced in Firmware v2.9 | 131 |
| 7.1 roAudioMetadata | 131 |
| 7.2 roImageMetadata | 132 |
| 8.0 Components First Introduced in Firmware v3.0 | 136 |
| 8.1 roSocketAddress | 136 |
| 8.2 roStreamSocket | 136 |
| 8.3 roDataGramSocket | 144 |

1.0 Introduction to the Roku Component Reference

Welcome to the Roku Streaming Player Component Reference. This document describes the BrightScript Components included with the SDK for use with the Roku Streaming Player. In addition to the components described in this manual, information on core components can be found in the BrightScript 2 Reference Manual. Together, these two documents provide details on the BrightScript programming language and the available API's. For a high-level overview of the development environment, please see the Roku Streaming Player Developer's Guide.

2.0 BrightScript Components

2.1 Introduction to BrightScript Components

BrightScript Components are the standardized way Roku exposes platform features to the SDK. Roku has made many of the core features of the platform available to SDK developers through the use of BrightScript Components. Developers can only access features of the Roku Streaming Player that have been explicitly made public through these interfaces. There is no way for 3rd party developers to extend the platform or add additional interfaces to BrightScript for use by the SDK.

2.2 BrightScript Component Interfaces and Methods

Every BrightScript Component consists of one or more "Interfaces". An RO Interface consists of one or more Methods. For example, the `roPosterScreen` has four interfaces: `ifPosterScreen`, `ifGetMessagePort`, `ifSetMessagePort`, and `ifHttpAgent`. The Interface `ifSetMessagePort` has one member: `SetMessagePort`.

For example:

```
p = CreateObject("roMessagePort")
posterscr = CreateObject("roPosterScreen")
springbrd = CreateObject("roSpringboardScreen")
posterscr.SetMessagePort(p)
springbrd.SetMessagePort(p)
```

This syntax makes use of a short cut provided by the language: The interface name is optional, unless it is needed to resolve name conflicts.

For example:

```
poster.SetMessagePort(p)
```

is the same as:

```
poster.ifSetMessagePort.SetMessagePort(p)
```

Note that the abstract Interface `ifSetMessagePort` is exposed and implemented by both the `roPosterScreen` and the `roSpringboardScreen` components. Once the method `SetMessagePort` is called, these components will send their events to the supplied message port. This is discussed more in the Event section below.

Roku intends to maintain backwards compatibility with any existing interfaces published in the SDK. Once an interface is defined and published, we may add methods to the interface but will

never change the signatures of existing methods. Any methods that are marked as deprecated in the SDK may be removed from the firmware at a future date.

2.3 Classes

A class name is the name used to create a BrightScript Component. For example, to create an `roUrlTransfer` object you would pass the class name to the `CreateObject()` routine:

```
video = CreateObject("roUrlTransfer")
```

2.4 Event Loops

When creating anything more than a very simple script, an Event Loop will need to be created. An Event Loop typically has this structure:

1. wait for the event
2. process the event
3. jump back to 1

Events are things like an on-screen button press, a remote button press, or a video that has finished playing back, etc. In this manual, we often refer to messages or events when describing the same concept.

By convention, BrightScript Component events work as follows.

- A BrightScript Component of type “`roMessagePort`” is created in BrightScript, by the user’s script.
- BrightScript Components that can send events are instructed to send their events to this message port. You could set up multiple message ports, and have each event go to its own message port, but it is usually simpler to just create one message port, and have the events all go to this one port. To instruct the BrightScript Component to send events to a specific port, use the *ifSetMessagePort* Interface.
- The script waits for an event. The actual function to do this is the `ifMessagePort.WaitMessage()`, but BrightScript also provides the built-in statement `WAIT` to make this easier.
- If multiple event types are possible, your script should determine which event was received during the wait, then process it. The script then jumps back to the wait.
- If you receive an unknown event in your event loop, you should ignore it; then just continue processing other events. Roku may occasionally add new events, and if your event is written to exit on unknown events, any future events that Roku may add will cause your application to misbehave.

A component-specific “event” can be generated by any BrightScript Component. The event arrives as a message at the message port associated with the component.

For example, the class “`roSpringboardScreen`” sends event messages of class “`roSpringboardScreenEvent`”. In general, a Brightscript function dealing with a specific component class should only react to events generated by that component with a filter like this in the event loop (example for an `roSpringboardScreen`) :

```
if type(message) = "roSpringboardScreenEvent"
```

Given the proper event class, one might be looking for a “button pressed” event type. One could test for this event type by calling `isButtonPressed()`. Any other event type predicates have similarly named methods. A typical test would be:

```
if message.isButtonPressed()
```

An event loop needs to be aware of the possible event classes and types it can get and process them.

The following is the list of tests for event types. If new types are added, these tests will remain valid.

```
Boolean isListItemSelected(Void)
Boolean isScreenClosed(Void)
Boolean isListFocused(Void)
Boolean isListSelected(Void)
Boolean isListItemFocused(Void)
Boolean isButtonPressed(Void)
Boolean isPlaybackPosition(Void)
Boolean isRemoteKeyPressed(Void)
Boolean isRequestSucceeded(Void)
Boolean isRequestFailed(Void)
Boolean isRequestInterrupted(Void)
Boolean isStatusMessage(Void)
Boolean isPaused(Void)
Boolean isResumed(Void)
Boolean isCleared(Void)
Boolean isPartialResult(Void)
Boolean isFullResult(Void)
Boolean isAdSelected(Void)
```

Since Firmware version 2.6:

```
Boolean isStorageDeviceInserted(Void)
Boolean isStorageDeviceRemoved(Void)
Boolean isStreamStarted(Void)
```

Since Firmware version 2.7:

```
Boolean isListItemInfo(Void)
Boolean isButtonInfo(Void)
```

The following methods allow you to obtain additional data from the event object:

```
Integer GetType(Void)
Integer GetIndex(Void)
String GetMessage(Void)
Integer GetData(Void) (Deprecated in v2.6)
```

Since Firmware version 2.6:

```
roAssociativeArray GetInfo(Void)
```

Since Firmware version 2.7:

In firmware v2.7, we added support for new remotes that have three new buttons:

- **Instant replay button.** Upon pressing this button, the video will skip back 7 seconds without re-buffering. Active only in video mode.
- **Info button.** This button must be programmed by app developers, or it will do nothing. App developers should provide contextually relevant information overlaid on video or other UI screens. Please notice the new predicates `isListItemInfo()` and `isButtonInfo()` listed above.
- **Back button.** Closes the current screen and pops the display stack in the UI. Active only in non-video mode. Developers must explicitly enable the Back button on the `roMessageDialog` component.

These buttons will require some support by developers in their event loops.

Example: Event Loops

```
port = CreateObject("roMessagePort")
screen = CreateObject("roSpringboardScreen")
screen.SetMessagePort(port)

while true
    msg = wait(0, port)
    if type(msg) = "roSpringboardScreenEvent" then
        if msg.isScreenClosed() then
            return -1
        elseif msg.isButtonPressed()
            print "button pressed: ";msg.GetIndex()
        endif
    endif
end while
```

2.5 Entry Points

In addition to the `Main()` BrightScript application entry point, firmware version 2.6 has added two additional entry points. `RunUserInterface()` is synonymous with `Main()` as a place for your application to start. `RunScreenSaver()` is a method that will be called when your script is registered as the system screen saver and the system has been idle and starts the screen saver.

```
Sub RunUserInterface(Void)
    ▪ Default script entry point for Roku applications.
Sub Main(Void)
    ▪ If there is no RunUserInterface() in the application, the method Main() will be called as the entry point for the application.
Sub RunScreenSaver(Void)
    ▪ Entered when the Roku has hit the configured idle-time. The screensaver to launch is selected by the user on the Screensaver settings page.
    ▪ The Roku box may not start the custom screensaver and will use a default screensaver instead if there is insufficient memory to start a screensaver slideshow
    ▪ roVideoPlayer and roAudioPlayer are not allowed to be run in a screensaver
    ▪ If you have any dialogs with ShowBusyAnimation() enabled, you'll want to make sure you close and kill the reference before painting a new screen on top of them. Otherwise, your app may lose focus when returning from the screensaver.
```

- Sub RunScreenSaverSettings(Void)
- Entered when the user selects “custom settings” on the Screensaver settings page and the application is configured as the system screensaver.

Since Firmware version 2.7:

- Sub RunUserInterface(Object aa)
- Can take an roAssociativeArray that can be passed via the Network Control API
- Sub Main(Object aa)
- Can take an roAssociativeArray that can be passed via the Network Control API

3.0 Roku Streaming Player Object Library

This section specifies each of the BrightScript Components that are included with Roku's consumer internet video players. In addition to the components in this document, core language components are documented in the BrightScript Reference Manual.

Summary of Components in the SDK

The following is a list of objects provided as part of the SDK.

Core Components

- roArray
- roByteArray
- roList
- roAssociativeArray
- roInt
- roString
- roFloat
- roBrSub
- roBoolean
- roInvalid
- roMessagePort
- roXMLElement
- roGlobal

Platform Components

- roDeviceInfo
- roUrlTransfer
- roUrlEvent
- roRegistry
- roRegistrySection
- roPinEntryDialog
- roPosterScreen
- roCodeRegistrationScreen
- roParagraphScreen
- roMessageDialog
- roOneLineDialog
- roVideoScreen
- roKeyboardScreen
- roSearchScreen
- roSearchHistory
- roSpringboardScreen
- roSlideShow
- roAudioPlayer
- roTimespan
- roDateTime
- roEVPCipher
- roEVPDigest

Platform Components Since v2.6

- roImageCanvas
- roFontRegistry
- roSystemLog
- roRegex
- roPath
- roFileSystem
- roVideoPlayer

Platform Components Since v2.7

- roGridScreen

Platform components Since v2.9

- roAudioMetadata
- roImageMetadata

Platform components Since v3.0

- roSocketAddress
- roStreamSocket
- roDataGramSocket

3.1 Roku Streaming Player Path Names

The Roku Streaming Player Pathnames have the following format:

<Device>:/dir1/dir2/.../<filename>.<ext>

<Device> can be one of:

tmp - temporary file storage device for the application

pkg - root of the application directory that provides read-only access to files provided in the pkg

Since Firmware version 2.6:

common – a common read-only filesystem that all plugins have access to.

Currently it only contains a CA certificate bundle that contains CA certs trusted by FireFox.

You are encouraged to use this file in your apps, especially if you are aggregating many different feeds. Please see the twitterOAuth sample application for a usage example.

common:/certs/ca-bundle.crt

ext1, ext2, ..., ext9 – storage devices recognized on the USB bus. Please see the usbplayer sample application for a usage example.

There is no concept of a current working directory or relative paths. All file system access should use the absolute Roku Streaming Player Pathname format above.

Pathnames have the following illegal character restrictions:

'<','>',';',':',',','\"', '/', '\\', '|', '?', '*', and any non-printable character.

You must use the Roku Streaming Player pathname convention for all pathnames in your script.

The registry is the only read/write persistent store available to applications. Temporary read/write file storage within a single run of an application is provided by the tmp: device. When the application exits, this temporary storage is purged. The pkg: device provides read-only access to

the application package directory, so your scripts can access any files shipped as part of your application.

A file path that does not specify a <Device> and or otherwise violates the file naming convention will cause the application to STOP and enter the Brightscript debugger.

Example: Roku Streaming Player path names in code snippets

```
theme.OverhangSliceSD = "pkg:/images/Overhang\_Slice\_SD43.png"

http.Http.GetToFile("tmp:/categorylist")

DeleteFile("tmp:/categorylist")

obj.SetCertificatesFile("pkg:/source/serverca.pem")

SDPosterUrl = "file:///pkg:/images/sd1932.jpg"
```

3.2 Application Directory and Manifest Data

Each application consists of a set of BrightScript source code files, images, and a manifest describing the application. The application is installed as a zip file containing the complete set of files.

The application manifest specifies a set of values and images that are displayed on the top-level menu before the application has been started. At system start-up, all of the installed applications on the system are enumerated and the top level menu is configured and displayed. These attributes are configured by setting the values in the application manifest file in the application directory. Note that each name=value pair must end with a newline character, or it will not be parsed by the firmware. Take care that the last line actually ends with a newline character.

The contents of the application manifest are:

```
major_version=1
minor_version=2
build_version=150
title=My Video Application
subtitle=providing the latest in cool videos
mm_icon_focus_hd=pkg:/images/mm_icon_focus_hd.png
mm_icon_side_hd=pkg:/images/mm_icon_side_hd.png
mm_icon_focus_sd=pkg:/images/mm_icon_focus_sd.png
mm_icon_side_sd=pkg:/images/mm_icon_side_sd.png
```

Each application provides four images for the main menu. A large, center-focus icon in both SD and HD sizes and a smaller non-focused side image in SD and HD. Images may be .png or .jpg files, but .png is required if alpha channel support is needed. See the sample applications for examples.

Table: Manifest Attributes

| Attribute | Type | Values | Example |
|-----------|--------|--------------|----------------------|
| title | String | Name of your | My Video Application |

| | | | |
|------------------|---------|---|---|
| | | application for display under the focus icon | |
| subtitle | String | Short promotional description of your application for display beneath the title | providing the latest in cool videos |
| major_version | Integer | Major portion of version string | 1 for version 1.2 |
| minor_version | Integer | Minor portion of version string | 2 for version 1.2 |
| build_version | Integer | Build number | 150 for the 150 th 1.2 build |
| mm_icon_focus_hd | String | Local URI for the large focused image for HD Size: 336 x 210 | pkg:/images/mm_icon_focus_hd.png |
| mm_icon_side_hd | String | Local URI for the small side image for HD Size: 108 x 69 | pkg:/images/mm_icon_side_hd.png |
| mm_icon_focus_sd | String | Local URI for the large focused image for SD Size: 248 x 140 | pkg:/images/mm_icon_focus_sd.png |
| mm_icon_side_sd | String | Local URI for the small side image for SD Size: 80 x 46 | pkg:/images/mm_icon_side_sd.png |

Since Firmware version 2.6:

| | | | |
|---------------------|---------|--|---------------------|
| screensaver_title | String | Name of your screensaver that will show up in the screensaver settings page | My cool screensaver |
| screensaver_private | Integer | If the value is '0', the screensaver_title will be selectable in the screensaver settings page. Otherwise, the screensaver will only run when your app is running. | 1 |

Since Firmware version 2.7:

| | | | |
|--------|---------|--|---|
| hidden | Integer | The hidden property in the manifest file tells the firmware to not display the app on the home screen. | 1 |
|--------|---------|--|---|

| | | | |
|--|--|--|--|
| | | Hidden apps can still be launched over the network via the External Control API. | |
|--|--|--|--|

Since Firmware version 2.9:

| | | | |
|------------------------|---------|---|---|
| requires_audiometadata | Integer | The roAudioMetadata requires the use of a dynamically loaded library that is not part of the initially booted image. Therefore, an entry must be added to the manifest of any applications that use the roAudioMetadata component so that it can be loaded when the channel is launched | 1 |
| requires_mkv | Integer | Playing MKV files requires the use of a dynamically loaded library that is not part of the initially booted image. Therefore, an entry must be added to the manifest of any applications that use the roAudioMetadata component so that it can be loaded when the channel is launched. | 1 |

3.3 Content Meta-Data

Content Meta-Data describes a viewable title that will be shown to the user. Content may be any supported type of video and the meta-data is used by the UI to format and display the title to the user. Some attributes (e.g. ContentType) affect how the title is displayed on screen, other attributes (e.g. SDPosterURL) specify where to fetch artwork to display with the content and other attributes (e.g. Title) are just rendered as text.

The content meta-data is stored in an associative array by the script and provided to the various screen objects as needed for display. In some cases an array of content meta-data may be provided so that the screen can render multiple items as a list. The attribute names are critical and used as the key to look up the attribute at runtime. The following table details the attributes in use. Certain attributes are recognized by particular screens, while others are more globally applicable. If the attribute is not a generally recognized attribute, the table below specifies where the attributes are recognized.

Table: Content Meta-Data Attributes

| Attribute | Type | Values | Example |
|----------------|---------|--|--|
| ContentType | String | Valid values = movie, episode, season, series, audio. Missing attribute = NotSet, | "movie" |
| Title | String | Content title as string. Episode title for series, movie title for films. | "The Dark Knight" |
| TitleSeason | String | Season name as string. Blank for movies. Season title for series. | "Battlestar Galactica Season 5" |
| Description | String | Description | "Batman, Gordon and Harvey Dent are forced to deal with the chaos ..." |
| Watched | Boolean | Flag indicating if movie has been watched. | True |
| Live | Boolean | Optional. Default false. Flag indicating video is live and replaces time remaining in progress bar to display "Live" | True |
| Url | String | URL of Image for roSlideShow or roImageCanvas | http://www.myco.com/img/vacation.jpg |
| SDBifUrl | String | URL for SD Trick Modes | http://www.myco.com/bif/sd1932.bif |
| HDBifUrl | String | URL for HD Trick Modes | http://www.myco.com/bif/hd1932.bif |
| SDPosterUrl | String | URL for SD Video Artwork | http://www.myco.com/img/sd1932.jpg |
| HDPosterUrl | String | URL for HD Video Artwork | http://www.myco.com/img/hd1932.jpg |
| StreamBitrates | roArray | Array of bitrates in kbps for content streams | [384, 500, 1000, 1500] |
| StreamUrls | roArray | Array of URL's for content streams | [" http://www.myco.com/vid/1932-1.mp4 ", " http://www.myco.com/vid/1932-2.mp4 ", " http://www.myco.com/vid/1932-3.mp4 ", " http://www.myco.com/vid/1932-4.mp4 "] |

| | | | |
|---------------------------|---------|--|---|
| StreamQualities | roArray | Array of Strings quality indicators identifying a stream as "SD" or "HD" | ["SD", "SD", "SD", "HD"] |
| StreamContentIDs | roArray | Array of String values logged in Roku logs to identify stream and bitrate played. | ["myco-19321-384.mp4", "myco-19321-500.mp4", "myco-19321-1000.mp4", "myco-19321-1500.mp4"] |
| StreamStickyHttpRedirects | roArray | Array of Boolean values indicating if the HTTP endpoint should be sticky and not subject to change on subsequent requests. Default = false | [false, false, false, false] |
| StreamStartTimeOffset | Integer | Optional. Default 0. The offset into the stream which is considered the beginning of playback. Time in seconds. | 3600 (e.g. 1 hour) |
| StreamFormat | String | Type of content Default: H.264/AAC in .mp4 Container | Valid values: mp4, wma, mp3 Note: mp4 will also accept .mov and .m4v files. Since Firmware version 2.6: "hls" Since Firmware version 2.9: These formats available over usb only: "mkv", "mka", "mks" Deprecated Since Firmware version 4.1: "wmv" |
| Length | Integer | Movie Length in Seconds | 3600 (e.g. 1 hour). Length zero displays at 0m, Length not set will not display. |
| BookmarkPosition | Integer | Last Play Position Bookmark in Secs | 1950 (e.g. 32 min 30 secs) |
| PlayStart | Integer | Optional Playback Start Offset in | 10 |

| | | | |
|-----------------------|---------|---|---|
| | | Seconds | |
| PlayDuration | Integer | Optional Playback Max Duration in Seconds | 120 (e.g. 2 minute preview) |
| ReleaseDate | String | Formatted Date String | "3/31/2009" |
| Rating | String | See Ratings Table | "PG-13" (e.g. value will display as artwork) |
| StarRating | Integer | Number 1-100 | 80 (e.g. 4 red stars displayed as artwork) Will also display fractional stars |
| UserStarRating | Integer | Number 1-100 | 80 (e.g. 4 yellow stars displayed as artwork) Will NOT display fractional stars |
| ShortDescriptionLine1 | String | Line 1 of Poster Screen Description | "The Dark Knight" |
| ShortDescriptionLine2 | String | Line 2 of Poster Screen Description | "Rent \$1.99, Buy \$14.99" |
| EpisodeNumber | String | Episode Number | "1" |
| Actors | roArray | List of Actor Names | ["Brad Pitt", "Angelina Jolie"] (array of names) |
| Actors | String | Individual Actor Name | "Brad Pitt" (single name as string) |
| Director | roArray | List of Director Names | ["Joel Coen", "Clint Eastwood"] (array of director names) |
| Director | String | Individual Director Name | "Christopher Nolan" (single name as string) |
| Categories | roArray | List of Category/Genre Names | ["Comedy", "Drama"] (array of genre names) |
| Categories | String | Individual Category/Genre Name | "Comedy" (single genre as string) |
| HDBranded | Boolean | Boolean indicating if HD branding should be displayed | True |
| IsHD | Boolean | Boolean indicating if content is HD | True |
| TextOverlayUL | String | roSlideShow displays this string in Upper Left corner of slide | "Joe's Photos" |
| TextOverlayUR | String | roSlideShow displays this string in Upper Right corner of slide | "3 of 20" |
| TextOverlayBody | String | roSlideShow displays this string on the bottom part of slide | "Wanda's 40'th Birthday" |

| | | | |
|--------|--------|--|-----------|
| Album | String | roSpringboard audio style uses this to display the album | "Achtung" |
| Artist | String | roSpringboard audio style uses to show artist | "U2" |

Since Firmware version 2.6:

| | | | |
|-------------------|--|---|---------------------------|
| SourceRect | Associative Array with params: {x:Integer y:Integer w:Integer h:Integer} | rolImageCanvas: The rectangle from the image that should be drawn. The default is the entire image at the origin. | {x:100,y:100,w:200,h:200} |
| TargetRect | Associative Array with params: {x:Integer y:Integer w:Integer h:Integer} | rolImageCanvas: The rectangle where the text/or image should be drawn. The default is the entire image at the origin. Can also be used with roFontMetrics to provide a target rect for the desired font size. Warning: BrightScript returns a Float by default when doing division. When calculating TargetRect values you may need an Integer conversion. | {x:400,y:200,w:200,h:200} |
| TargetTranslation | Associative Array with params: {x,y} | rolImageCanvas: The amount to translate the coordinate system prior to drawing the image and/or text. Translation is done before rotation. Default: {0,0} | {x:100,y:100} |
| TargetRotation | Float | rolImageCanvas: The angle (in degrees) to rotate the coordinate system prior to drawing the image and/or text. Default: 0 | 90.0 |

| | | | |
|-----------------|--------------------|--|---|
| CompositionMode | String | rolImageCanvas: Either " Source " (where source pixels completely replace destination pixels) or " Source_Over " (where source pixels are alpha blended with destination pixels). | "Source_Over" |
| Text | String | rolImageCanvas: The text is drawn into the TargetRect. If it doesn't fit, then it will be clipped. | "Hello ImageCanvas" |
| TextAttrs | roAssociativeArray | Please see Table3: TextAttrs for the key description details | { Color:"#FFCCCC", Font:"Medium", HAlign:"HCenter", VAlign:"VCenter", Direction:"LeftToRight" } |
| SubtitleUrl | String | roVideoPlayer and roVideoScreen: Will display an overlay of subtitles stored in an SRT formatted file. | " http://www.myco.com/vid/1932.srt " |

| | | | |
|---------|--|--|---|
| Stream | roAssociativeArray with Params: { url : String bitrate : Integer quality : Boolean contentid : String stickyredirects : Boolean } | roVideoPlayer and roVideoScreen: roArray that has parameters representing the stream settings that were set as individual roArrays in previous firmware revisions. The old method is still supported and descriptions of the parameters can be found under those content- meta data entries. For url please see StreamUrls, for quality it is now a Boolean that is true for hdQuality. | { url:" http://me.com/big.m3u8 ", quality : true contentid : "big-hls" } |
| Streams | roArray of roAssociativeArrays with Params: [{ url : String bitrate : Integer quality : Boolean contentid : String stickyredirects : Boolean }, {} ...] | roVideoPlayer and roVideoScreen: roArray of roAssociativeArrays that have parameters representing the stream settings that were set as individual roArrays in previous firmware revisions. The old method is still supported and descriptions of the parameters can be found under those content- meta data entries. For url, please see StreamUrls, for quality it is now a Boolean that is true for hdQuality. | [{ url : " http://me.com/x-384.mp4 ", bitrate : 384 quality : false contentid : "x-384" }, { url : http://me.com/x-2500.mp4 , bitrate : 2500 quality : true contentid : "x-1500" }] |

Since Firmware version 2.7:
















| | | | |
|--------------|---------|--|------|
| MinBandwidth | Integer | roVideoPlayer or roVideoScreen: Will only select variant streams with a bandwidth higher than this minimum bandwidth. Only has an effect on HLS streams. Default is 250 so that audio only variant streams are not considered by default. By default Wowza servers set streams to 64 kbs, so you might want to set this parameter to something smaller than 64 when first testing Wowza streams. You will eventually want to specify the Wowza bitrates with a smil file (Please see the encoding guide). Units are Kbps. | 48 |
| MaxBandwidth | Integer | roVideoPlayer or roVideoScreen: Will only select variant streams with a bandwidth less than this maximum bandwidth. Units are Kbps. Typically used only in testing. | 2500 |
| AudioPIDPref | Integer | roVideoPlayer or roVideoScreen If the specified preferred PID audio stream exists, it will be chosen. Otherwise the last audio stream will be chosen. | 483 |
| FullHD | Boolean | roVideoPlayer or roVideoScreen Specify that this stream was encoded at 1080p resolution. | true |
| FrameRate | Integer | roVideoPlayer or roVideoScreen Specify the 1080p stream was encoded at 24 or 30 fps. | 24 |

Since Firmware version 2.8:

| | | | |
|-------------------|--------|--|-----------------|
| SwitchingStrategy | String | <p>roVideoPlayer or roVideoScreen: Specify different stream switching algorithms to be used in HLS adaptive streaming. Only has an effect on HLS streams. Default is "no-adaptation". We recommend staying with the default unless the default seems to be problematic with your streams.</p> <p>Possible strategy values: "unaligned-segments". This strategy does not require segments to align across variants. Also does not switch as often as full-adaptation.</p> <p>"full-adaptation" uses measured bandwidth and buffer fullness to determine when to switch. This strategy requires that segments align across variants as the HLS spec requires. This is the new default.</p> <p>"minimum-adaptation" Uses only measured bandwidth in determining switching strategy. This strategy requires that segments align across variants as the HLS spec requires.</p> <p>"no-adaptation" Only switches bitrates on a rebuffer.</p> | full-adaptation |
|-------------------|--------|--|-----------------|

Table 2: MPAA and TV Ratings

The Rating attribute contains the MPAA or TV rating stored as a string. At runtime, the ratings are shown with an icon instead of rendering the string as text. The following table shows the list of valid values for the Rating attribute and the resulting icon that will be displayed for each value.

| Rating Attribute Value | Result |
|------------------------|---|
| G |  |
| NC-17 |  |
| PG |  |
| PG-13 |  |
| R |  |
| UR |  |
| UNRATED |  |
| NR |  |
| TV-Y |  |
| TV-Y7 |  |
| TV-Y7-FV |  |
| TV-G |  |
| TV-PG |  |
| TV-14 |  |
| TV-MA |  |

Since Firmware version 2.6:

Table 3: TextAttrs

| | | | |
|--------|--------|--|-------------|
| Color | String | roImageCanvas (applies to text): Color of the text. The color may also include "alpha" channel bits in the color value 00=transparent FF=opaque. The alpha channel enables blending the background with the images. Default: "#FFFFFF" | "#FF0033FF" |
| Font | String | roImageCanvas (applies to text): "Small", "Medium", "Large", or "Huge". Default: "Medium" Also can use any fonts registered by the roFontRegistry Object and returned by its Get() method | "Large" |
| HAlign | String | roImageCanvas (applies to text): Controls the vertical alignment of the text in | "Right" |

| | | | |
|---------------|--------|--|---------------|
| | | the TargetRect. Options are: "Left", "Center", "Right", "Justify" Default: "Center" | |
| VAlign | String | rolmageCanvas (applies to text): Controls the vertical alignment of the text in the TargetRect. Options are: "Top", "Middle", "Bottom" Default: "Middle" | "Bottom" |
| TextDirection | String | rolmageCanvas (applies to text): "LeftToRight" or "RightToLeft" Default: "LeftToRight" | "RightToLeft" |

4.0 Original Platform Components (Since v2.4)

The following section provides details for all the platform specific components first introduced in Roku Firmware version 2.4. The core components are documented in the BrightScript 2 Reference Manual. For each class a brief description is given, a list of interfaces, and details on the member functions in the interfaces. As we add methods to components, we will group them in sections indicating the firmware version where they first showed up.

4.1 General Attributes

All full-screen SDK UI components share a common set of attributes when displayed. The title area or “overhang” is designed to provide a uniform color scheme for an application. The color and logo may be set by the developer by providing the proper art assets for the application. The concept of breadcrumbs allows the title area to provide navigational guideposts as a reference to the current and previous screen.

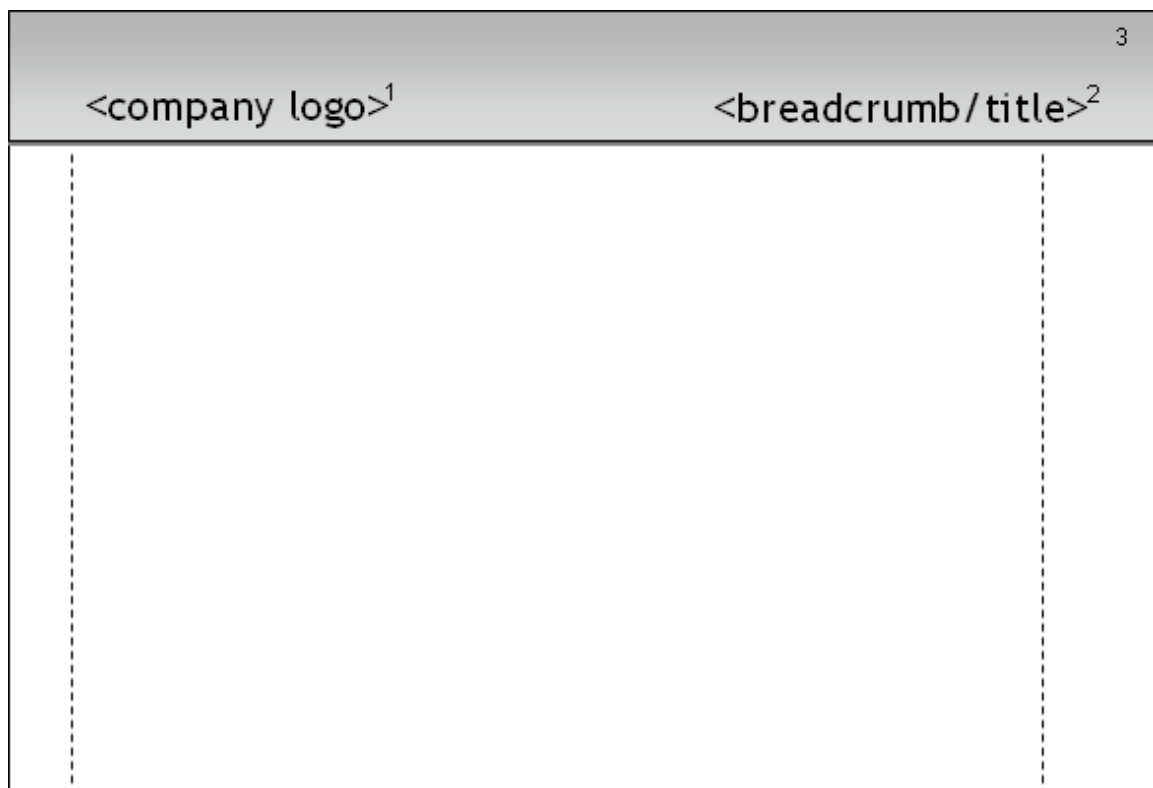


Diagram: Generic screen attributes

Notes:

1. The company logo is provided as a .png (portable network graphic) resource file that is distributed as part of the application package.
2. Depending on the type of screen, the <breadcrumb/title> area may display either navigational breadcrumbs or a title string. Breadcrumbs are based on strings that describe the current and previous location and automatically formatted by the screen to appear in a standard style.

3. The header area or “overhang” color may be set by providing a new “slice” graphic (.png). The slice represents a small portion of the overhang which is replicated across the entire title area.
4. The suggested maximum Overhang height is 83 pixels for SD and 124 pixels for HD. The enforced maximum Overhang height is 92 for SD and 138 for HD. Between the suggested maximum height and the enforced maximum height is a transition area in which application developers should alpha blend their Overhang art into the screen background.
5. Art rendered in the Overhang transition area (83 - 92 px for SD and 124 - 138 px for HD) will be rendered over by the Filter Banner, Modal Dialogs, and other native UI widgets. Application developers should not assume that the art in the Overhang transition zone will always be viewable by the end user.

4.2 roAppManager

The Application Manager provides a set of API's to set application level attributes. These attributes are currently focused on configuring the look-and-feel of your application. The use of screen styles gives each application a consistent look-and-feel, but attributes such as colors, fonts and logos need to be unique for each application. Establishing a set of artwork and colors allows the developer to specify a theme for their application. If these values are not set, the application will use the default theme values.

The roAppManager implements the following interfaces:

Interface: ifAppManager

```
Void SetTheme(Object attributeArray)
    ▪ Set a group of theme attributes for the application as a set. The attributeArray provided is an roAssociativeArray of attribute/value pairs for each attribute being set. The roAssociativeArray may be populated manually at runtime with values or the developer can provide and parse an XML file containing the theme using the roXMLElement object. Existing values for attributes will be overwritten by the values provided. Any values set by a previous SetTheme or SetThemeAttribute call, but not included in the array currently provided by with the subsequent call will remain unchanged.
```

```
Void SetThemeAttribute(String attributeName, String attributeValue)
    ▪ Set an individual theme attribute for the application. The attributeName provided is the name of one of the settable theme attributes and the value is the desired setting. This value will override the default value for that attribute or modify the value provided by a previous SetTheme or SetThemeAttribute call to the new value provided. If the attributeName is not valid, no action is performed.
```

```
Void ClearThemeAttribute(String attributeName)
    ▪ Clears a previously set attribute and reverts to using the default value for that attribute.
```

Since Firmware version 2.6:

```
roTimespan GetUptime(Void)
    ▪ Returns an roTimespan object which is “marked” when the user clicked on the application button on the home screen. Calling TotalMilliseconds() on the returned roTimespan object returns the total number of milliseconds since the application started.
```

Table: Theme Attributes

| Attribute | Type | Values | Example |
|-----------------|--------|-------------|-----------------------------|
| OverhangSliceSD | String | URI for the | pkg:/images/overhang_sd.png |

| | | | |
|---------------------------------|------------------|---|-----------------------------|
| | | overhang slice (thin piece of top of screen border). | |
| OverhangPrimaryLogoSD | String | Small application logo formatted for display in overhang top left. | pkg:/images/co_logo_sd.png |
| OverhangPrimaryLogoOffsetSD_X | Number as String | Offset in pixels from the top-left origin of the display. Range 0 to 720. | "25" |
| OverhangPrimaryLogoOffsetSD_Y | Number as String | Offset in pixels from the top-left origin of the display. Range 0 to 480. | "50" |
| OverhangSecondaryLogoSD | String | Small application logo formatted for display in overhang top left. | pkg:/images/co_logo_sd.png |
| OverhangSecondaryLogoOffsetSD_X | Number as String | Offset in pixels from the top-left origin of the display. Range 0 to 720. | "25" |
| OverhangSecondaryLogoOffsetSD_Y | Number as String | Offset in pixels from the top-left origin of the display. Range 0 to 480. | "50" |
| OverhangSliceHD | String | URI for the overhang slice (thin piece of border at the top of the screen in HD size) | pkg:/images/overhang_sd.png |

| | | | |
|---------------------------------|------------------|--|----------------------------|
| OverhangPrimaryLogoHD | String | Small application logo formatted for display in overhang top left. | pkg:/images/co_logo_sd.png |
| OverhangPrimaryLogoOffsetHD_X | Number as String | Offset in pixels from the top-left origin of the display. Range 0 to 1280 | "25" |
| OverhangPrimaryLogoOffsetHD_Y | Number as String | Offset in pixels from the top-left origin of the display films. Range 0 to 720 | "50" |
| OverhangSecondaryLogoHD | String | Small application logo formatted for display in overhang top left. | pkg:/images/co_logo_sd.png |
| OverhangSecondaryLogoOffsetHD_X | Number as String | Offset in pixels from the top-left origin of the display. Range 0 to 1280 | "25" |
| OverhangSecondaryLogoOffsetHD_Y | Number as String | Offset in pixels from the top-left origin of the display films. Range 0 to 720 | "50" |
| ButtonMenuHighlightText | String | HTML HEX Color Value | #0033FF |
| ButtonMenuNormalOverlayText | String | HTML HEX Color Value | "#B0B0B0" |
| ButtonMenuNormalText | String | HTML HEX Color Value | "#686868" |
| ButtonNormalColor | String | HTML HEX Color Value | "#FF00FF" |
| ButtonHighlightColor | String | HTML HEX Color Value | "#FF00FF" |
| BackgroundColor | String | HTML HEX | "#E0DFDF" |

| | | | |
|-------------------------------|--------|--|--|
| | | Color Value | |
| ParagraphBodyText | String | HTML HEX Color Value | "#FF00FF" |
| ParagraphHeaderText | String | HTML HEX Color Value | "#FF00FF" |
| PosterScreenLine1Text | String | HTML HEX Color Value | "#FF00FF" |
| PosterScreenLine2Text | String | HTML HEX Color Value | "#FF00FF" |
| BreadcrumbTextLeft | String | HTML HEX Color Value | "#FF00FF" |
| BreadcrumbTextRight | String | HTML HEX Color Value | "#FF00FF" |
| BreadcrumbDelimiter | String | HTML HEX Color Value | "#FF00FF" |
| RegistrationCodeColor | String | HTML HEX Color Value | "#FF00FF" |
| RegistrationFocalColor | String | HTML HEX Color Value | "#FF00FF" |
| SpringboardTitleText | String | HTML HEX Color Value | "#FF00FF" |
| SpringboardActorColor | String | HTML HEX Color Value | "#FF00FF" |
| SpringboardSynopsisColor | String | HTML HEX Color Value | "#FF00FF" |
| SpringboardSynopsisText | String | HTML HEX Color Value | "#FF00FF" |
| SpringboardGenreColor | String | HTML HEX Color Value | "#FF00FF" |
| SpringboardRuntimeColor | String | HTML HEX Color Value | "#FF00FF" |
| SpringboardDirectorLabelColor | String | HTML HEX Color Value | "#FF00FF" |
| SpringboardDirectorColor | String | HTML HEX Color Value | "#FF00FF" |
| SpringboardDirectorLabel | String | Director Label | "Written By" |
| SpringboardArtistColor | String | HTML HEX Color Value | "#FF00FF" |
| SpringboardArtistLabelColor | String | HTML HEX Color Value | "#FF00FF" |
| SpringboardAlbumColor | String | HTML HEX Color Value | "#FF00FF" |
| SpringboardAlbumLabelColor | String | HTML HEX Color Value | "#FF00FF" |
| SpringboardArtistLabel | String | Artist Label | "by" |
| SpringboardAlbumLabel | String | Album Label | "on" |
| EpisodeSynopsisText | String | HTML HEX Color Value | "#FF00FF" |
| FilterBannerSliceSD | String | URL to set SD Filter Banner Background | "pkg:/images/Filter_BackgndSlice_SD43.png" |

| | | | |
|---------------------------|--------|--|--|
| | | Image | |
| FilterBannerActiveSD | String | URL to set SD Filter Banner Active/Focus Highlighter | "pkg:/images/Filter_ActiveHint_SD43.png" |
| FilterBannerInactiveSD | String | URL to set SD Filter Banner Inactive Highlighter | "pkg:/images/Filter_InactiveHint_SD43.png" |
| FilterBannerSliceHD | String | URL to set HD Filter Banner Background Image | "pkg:/images/Filter_BackgndSlice_SD43.png" |
| FilterBannerActiveHD | String | URL to set HD Filter Banner Active/Focus Highlighter | "pkg:/images/Filter_ActiveHint_SD43.png" |
| FilterBannerInactiveHD | String | URL to set HD Filter Banner Inactive Highlighter | "pkg:/images/Filter_InactiveHint_SD43.png" |
| FilterBannerActiveColor | String | HTML HEX Color Value | "#FF00FF" |
| FilterBannerInactiveColor | String | HTML HEX Color Value | "#FF00FF" |
| FilterBannerSideColor | String | HTML HEX Color Value | "#FF00FF" |

Since Firmware version 2.7:

| | | | |
|-----------|--------|---|----------------|
| ThemeType | String | Theme type that currently applies to dialogs and the keyboard screen. Generic-dark is the only valid value. Otherwise the default theme applies | "generic-dark" |
|-----------|--------|---|----------------|

| | | | |
|---------------------------|------------------|--|----------------------------------|
| GridScreenOverhangSliceSD | String | URI for the overhang slice (thin piece of top of screen border). | "pkg:/images/gridoverhangSD.png" |
| GridScreenLogoSD | String | Logo formatted for display in the overhang | "pkg:/images/gridlogoSD.png" |
| GridScreenLogoOffsetSD_X | Number as String | Offset in pixels from the top-left origin of the display. Range 0 to 720. | "324" |
| GridScreenLogoOffsetSD_Y | Number as String | Offset in pixels from the top-left origin of the display. Range 0 to 480. | "21" |
| GridScreenOverhangSliceHD | String | URI for the overhang slice (thin piece of top of screen border). | "pkg:/images/gridoverhangHD.png" |
| GridScreenLogoHD | String | Logo formatted for display in the overhang | "pkg:/images/gridlogoHD.png" |
| GridScreenLogoOffsetHD_X | Number as String | Offset in pixels from the top-left origin of the display. Range 0 to 1280. | "592" |
| GridScreenLogoOffsetHD_Y | Number as String | Offset in pixels from the top-left origin of the display. Range 0 to 720. | "31" |
| GridScreenBackgroundColor | String | HTML HEX Color Value Must be a grayscale value | "#363636" |

| | | | |
|------------------------------------|--------|---|-----------|
| GridScreenMessageColor | String | HTML HEX Color Value. Must be a grayscale value | "#808080" |
| GridScreenRetrievingColor | String | HTML HEX Color Value. Must be a grayscale value | "#CCCCCC" |
| GridScreenListNameColor | String | HTML HEX Color Value. Must be a grayscale value | "#FFFFFF" |
| GridScreenDescriptionTitleColor | String | HTML HEX Color Value. | "#00FFFF" |
| GridScreenDescriptionDateColor | String | HTML HEX Color Value. | "#FF005B" |
| GridScreenDescriptionRuntimeColor | String | HTML HEX Color Value. | "#5B005B" |
| GridScreenDescriptionSynopsisColor | String | HTML HEX Color Value. | "#606000" |
| CounterTextLeft | String | HTML HEX Color Value. | "#FF0000" |
| CounterSeparator | String | HTML HEX Color Value. | "#00FF00" |
| CounterTextRight | String | HTML HEX Color Value. | "#0000FF" |

Since Firmware version 2.8:

| | | | |
|--------------------------|--------|---|-------------------------------------|
| GridScreenFocusBorderSD | String | URL to set SD Focus image on Active Grid Poster | "pkg:/images/Border_16x9_SD 43.png" |
| GridScreenFocusBorderHD | String | URL to set SD Focus image on Active Grid Poster | "pkg:/images/Border_16x9_HD .png" |
| GridScreenBorderOffsetSD | String | String representing point "(x, y)" that is the offset from the upper left corner of the focused SD image. Set to the negative width & | "(-20,-20)" |

| | | | |
|-------------------------------|--------|--|---|
| | | height of border. | |
| GridScreenBorderOffsetHD | String | String representing point “(x, y)” that is the offset from the upper left corner of the focused HD image. Set to the negative width & height of border. | "(-25,-25)" |
| GridScreenDescriptionImageSD | String | URL to set SD Description callout background image on Grid. | "pkg:/images/Description_Background_SD43.png" |
| GridScreenDescriptionImageHD | String | URL to set SD Description callout background image on Grid. | "pkg:/images/Description_Background_HD.png" |
| GridScreenDescriptionOffsetSD | String | String representing point “(x, y)” that is the offset from the upper left corner of the focused SD image. Negative values have the description above and to the left of the focused image. | "(125,170)" |
| GridScreenDescriptionOffsetHD | String | String representing point “(x, y)” that is the offset from the upper left corner of the focused | "(190,255)" |

| | | | |
|----------------------------|------------------|--|------|
| | | SD image. Negative values have the description above and to the left of the focused image. | |
| GridScreenOverhangHeightSD | Number as String | The SD overhang height. Default: "49" | "55" |
| GridScreenOverhangHeightHD | Number as String | The HD overhang height. Default: "69" | "75" |

Example Code: roAppManager setting theme values

```

Sub SetApplicationTheme()

    app = CreateObject("roAppManager")
    theme = CreateObject("roAssociativeArray")

    theme.OverhangOffsetSD_X = "72"
    theme.OverhangOffsetSD_Y = "25"
    theme.OverhangSliceSD = "pkg:/images/Overhang_Slice_SD43.png"
    theme.OverhangPrimaryLogoSD = "pkg:/images/Logo_Overhang_
SD43.png"
    theme.OverhangOffsetHD_X = "123"
    theme.OverhangOffsetHD_Y = "48"
    theme.OverhangSliceHD = "pkg:/images/Overhang_Slice_ HD.png"
    theme.OverhangPrimaryLogoHD = "pkg:/images/Logo_Overhang_HD.png"

    app.SetTheme(theme)

End Sub

```

Since Firmware version 3.0:

| | | | |
|-----------------|--------|---|-----------|
| DialogTitleText | String | HTML HEX Color Value Must be a grayscale value | "#363636" |
| DialogBodyText | String | HTML HEX Color Value. Must be a grayscale value | "#808080" |

4.3 roPosterScreen

The Poster Screen provides a graphical display of poster art for content selection or can be used as a submenu to provide hierarchical structure to the application. In some cases, applications may wish to present a flat single-level list of titles in a queue, but the Poster Screen can also be used at multiple levels in the application to provide hierarchical browsing. It also provides an optional “filter banner” for displaying categories representing filtered subsets of the data or categorized groups.

Each item in the poster screen is represented by an image (aka poster), so any type of item that can be visually represented by an image can be displayed in the poster screen. It is used to show lists of data to users and common patterns include content categories, movies, podcasts, and search results.

Just below the overhang is the filter banner. It allows a method of easily selecting or filtering content based on categories. The categories are set by the developer during screen initialization, and the script is notified when a new category is highlighted or selected. Based on the event notification, the script can set the desired content in the view. The filter banner is optional.

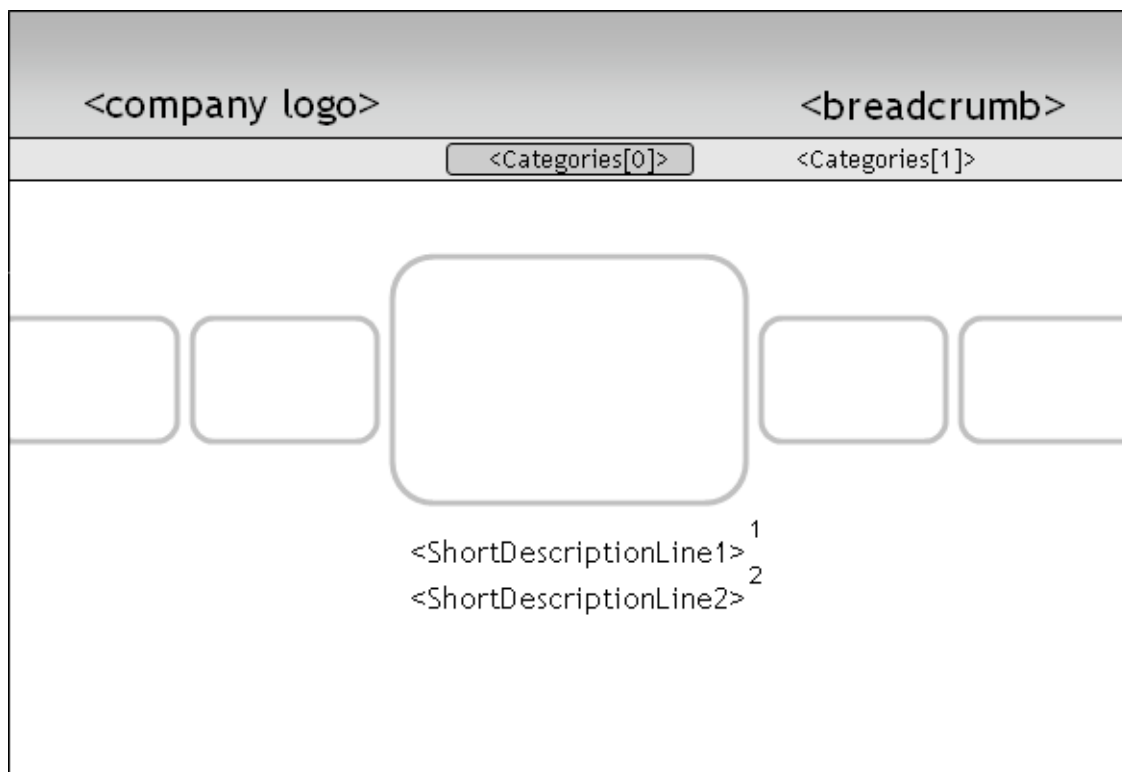


Diagram: roPosterScreen (flat-category)

Notes:

1. ShortDescriptionLine1 from the content metadata. Generally the category title.
2. ShortDescriptionLine2 from the content metadata. Generally a description for the category.

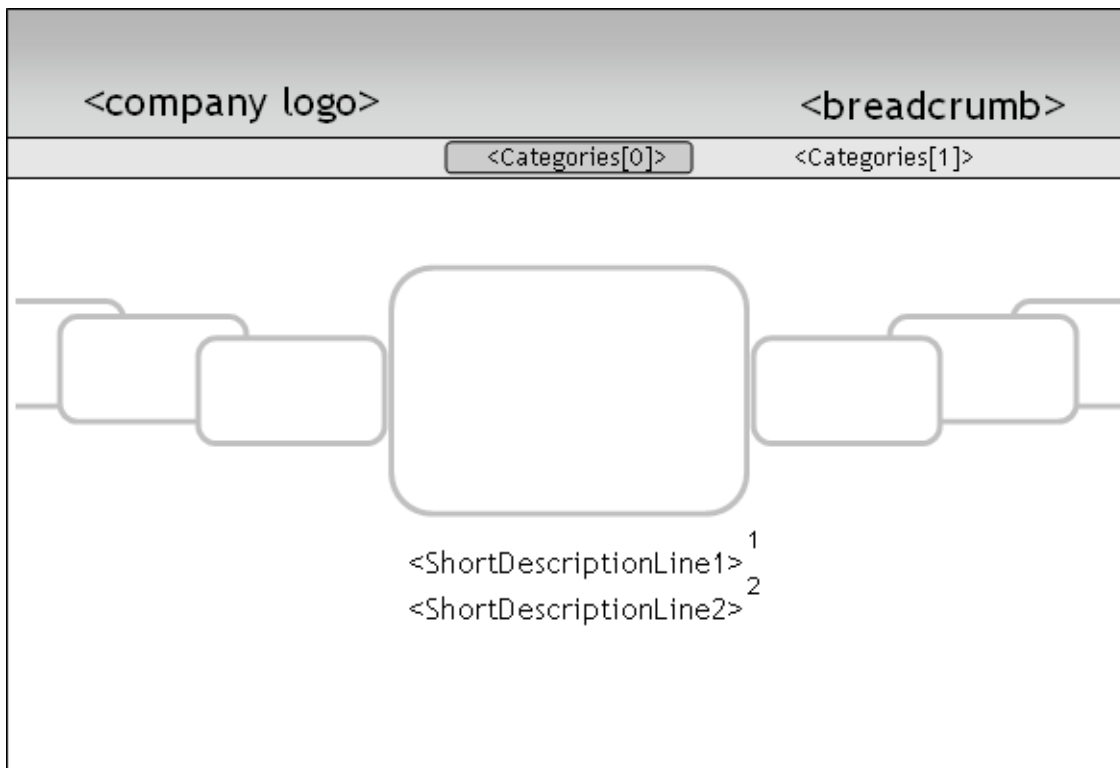


Diagram: roPosterScreen (arced-landscape)

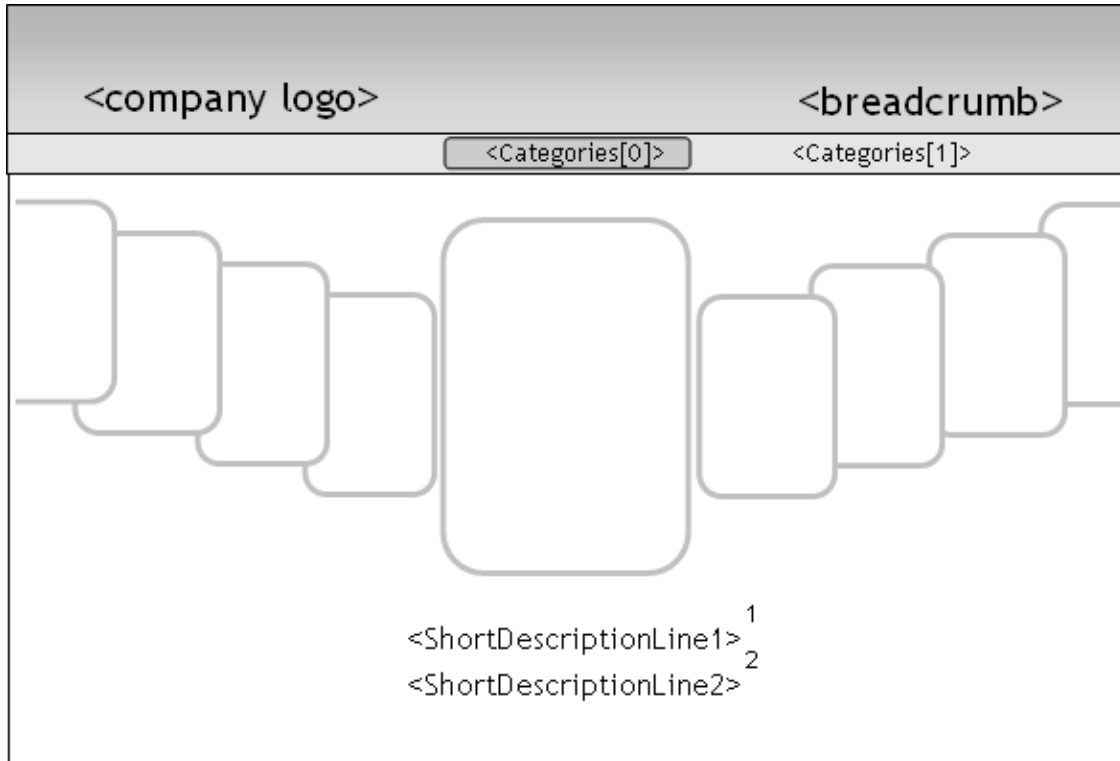


Diagram: roPosterScreen (arced-portrait)

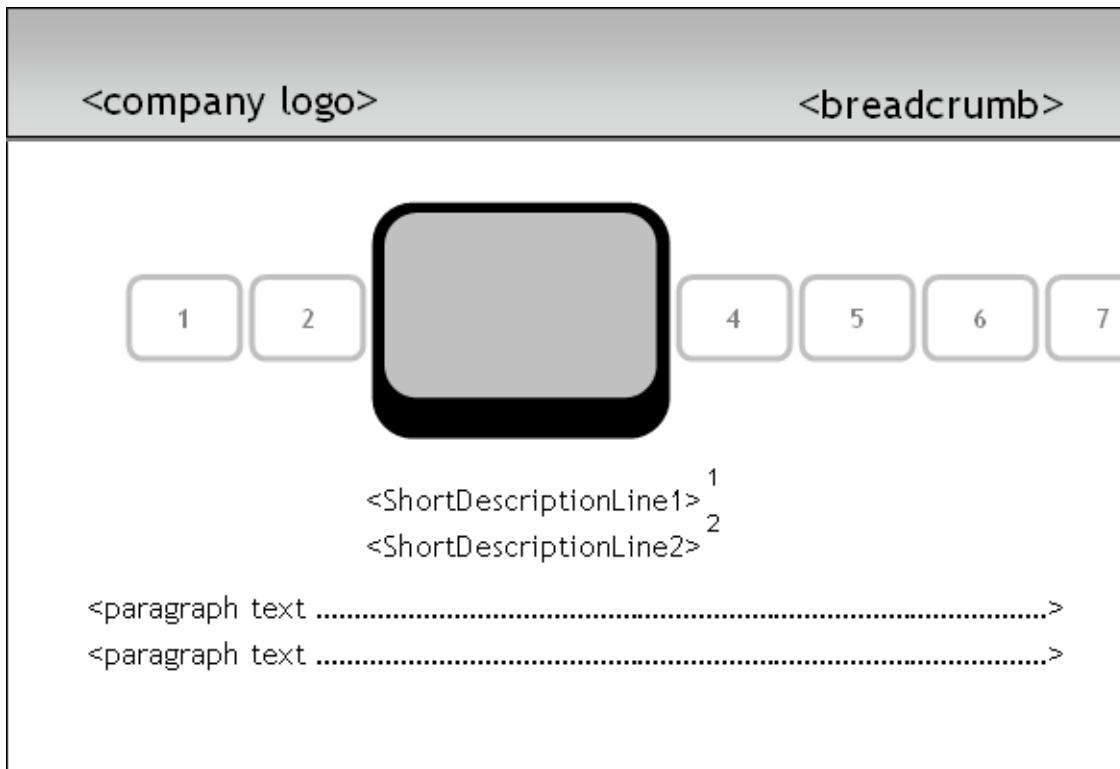


Diagram: `roPosterScreen` (flat-episodic)

Notes:

1. TV content is often displayed as a series of episodes within a season. The flat-episodic screen type provides a standard way to display episodic content, such as a TV series.
2. There is also a flat-episodic-16x9-episodic screen type to display episodic content with 16x9 images.
3. The paragraph text allows the user to view the synopsis for the currently selected episode. As the user scrolls right/left to select a new episode, the paragraph text and the short description lines are updated to reflect the description of the highlighted episode.
4. In order to see poster art in the side posters instead of episode numbers, please ensure that the `SDPosterUrl` and `HDPosterUrl` are defined for the content and that `episodeNumber` is not defined for that content. `EpisodeNumber` overrides the poster URL.

This object is created with no parameters:

- `CreateObject("roPosterScreen")`

The `roPosterScreen` implements the following interfaces:

Interface: `ifPosterScreen`

Void `SetContentList(Object contentList)`

- Set the list of content to be displayed by the screen. The caller passes an `roArray` of `roAssociativeArrays` (Content Meta-Data objects) representing the information for each title to be displayed on screen. See "Content Meta-Data" for details on the attributes for each element in the array. The screen is responsible for fetching the poster art from the URL's specified and all user navigation within the list.

```
Object GetContentList(Void)
```

- Returns the roArray of content meta-data passed via the SetContentList call.

```
Void SetListNames(Object names)
```

- Sets the list of categories to be displayed in the filter banner at the top of the poster screen. The caller passes the list as an array of Strings. Each String represents a new category to be displayed at the top. The display order is the same as the order of the categories in the array passed by the caller.

```
Void SetListStyle(String style)
```

- Set the display style for the poster screen. Styles allow the poster screen to look differently for different types of content or different usage. The following is the list of valid screen styles:
 - “arced-portrait” – arced list of poster art displayed in portrait (tall) style for DVD/Movie poster artwork. This is the default poster screen display format. Artwork sizes: SD=158x204; HD=214x306
 - “arced-landscape” - arced list of poster art displayed in landscape (wide) style for 4:3 aspect ratio TV artwork. Artwork sizes: SD=214x144; HD=290x218
 - “arced-16x9” – arced list of poster art displayed in 16x9 aspect ratio. Artwork sizes: SD=285x145; HD=385x218
 - “arced-square” – Arced list of poster art displayed in a square frame style. Artwork sizes: SD = 223x200; HD = 300x300 Note: is non-square NTSC pixel aspect ratio images
 - “flat-category” – flat (non-arced) list of categories. This is generally used for the providers top level menu screen. Artwork sizes are: SD=224x158; HD=304x237
 - “flat-episodic” – flat (non-arced) list of episodes for series content such as a TV show. Artwork sizes: SD=166x112; HD=224x168
 - “rounded-rect-16x9-generic” – 16x9 poster style. Artwork Sizes SD=177x90; HD=269x152
 - “flat-episodic-16x9” – flat (non-arced) list of episodes for series content such as a TV show. Artwork sizes: SD=185x94; HD=250x141

```
Void SetListDisplayMode(String displayMode)
```

- Sets the mode for displaying images in the poster screen. This allows images to be either scaled to completely fill the poster frame (scale-to-fill) or scaled to fit inside the poster frame (scale-to-fit) while maintaining aspect ratio. Valid display modes are:
 - scale-to-fill – scale image to completely fill the rectangle of the bounding frame (Default)
 - scale-to-fit. – scale image to fit horizontally or vertically as appropriate while still maintaining aspect ratio. Note that scale-to-fit may result in pillar-box or letter-box display of images.
 - zoom-to-fill – scales and crops image to maintain aspect ratio and completely fill the rectangle of the bounding frame.
 - photo-fit – Uses several methods to fit the image with a different aspect ratio to the screen. First, it will asymmetrically scale up to a maximum of 5%. Second, for landscape images, if vertical cropping is necessary, it will remove two lines off the bottom for every one line off the top up to a maximum of 30% of the image. For all images, if horizontal cropping is necessary it will crop an equal amount from both sides.

```
Void SetFocusedList(Integer itemIndex)
```

- Zero-based index of item in filter banner to be given focus. The selected item is displayed in the center of the filter banner and highlighted to designate focus.

```
Void SetFocusedListItem(Integer itemIndex)
```

- Zero-based index of item in poster list to be selected. The selected item is displayed in the center of the screen and bordered to designate focus.

```
Void SetBreadcrumbText(String location1, String location2)
```

- Breadcrumbs allow the application to display a two-part navigational title which shows the current and the previous locations in the application hierarchy (e.g. TV – Friends). If both location values are set, the application will display the title in breadcrumb format. If only

the first location is set, the application will display the specified text in the title area like the `SetTitle` API call.

```
Void SetBreadcrumbEnabled(Boolean enable)
    ▪ Show or hide the breadcrumb text in the title area.
Void ShowMessage(String message)
    ▪ Displays a semi-transparent popup message box to the user in the center of the screen over the poster screen. Generally used for error messages.
Void ClearMessage(Boolean clear)
    ▪ Clears the message from the previous ShowMessage call.
Void SetAdURL(String sdAdURL, String hdAdURL)
    ▪ Set the URL of the banner ad image to be displayed on the poster screen. This is currently only valid for the “arced-landscape” and “flat-category” style of poster screens. Banner ad sizes are as follows:
      Banner Ad HD = 728x90 Banner Ad SD = 540x60
Void SetAdSelectable(Boolean isSelectable)
    ▪ Sets the banner ad to be selectable or display only. By default the banner ad is not selectable. When enabled the user can navigate and move focus to the banner image. When selected, the screen will receive an roPosterScreenEvent and the msg will return true for isAdSelected().
Void SetAdDisplayMode(String displayMode)
    ▪ Sets the scale mode for displaying ad images on the poster screen. The ad display is only available on “arced-landscape” and “flat-category” list styles. Valid display modes are:
      ○ scale-to-fill – scale image to completely fill the rectangle of the bounding frame (default)
      ○ scale-to-fit. – scale image to fit horizontally or vertically as appropriate while still maintaining aspect ratio. This is the preferred display mode for ads.
Boolean Show(Void)
    ▪ Display or refresh the screen after initial creation or state changes.
Void Close(Void)
    ▪ Close the screen and delete the associated object. Useful for avoiding screen flicker when the display order of your screens does not resemble a stack.
Void SetTitle(String title)
    ▪ Set the title for the screen to the specified string.
```

Since Firmware version 3.0:

```
Void SetFocusToFilterBanner(Boolean enable)
    ▪ When enable is true, set focus to filter banner.
    ▪ When enable is false, set focus to posters.
```

Interface: `ifHttpAgent`

The `ifHttpAgent` methods setup the way URLs in the Content Meta-Data are accessed via this object.

```
Boolean AddHeader(String name, String value)
    ▪ Add the specified HTTP header.
    ▪ If “x-roku-reserved-dev-id” is passed as name, the Roku box ignores the passed in value. In its place, sends the devid of the application publisher of the currently running app. This allows the developer’s server to know which client app is talking to it.
    ▪ Developers may set any headers except what’s explicitly reserved by Roku. The Roku box will enforce ownership of values in the namespace x-roku-reserved-
```

- The Roku box will set initial values of well known headers (e.g. User-Agent, Content-Length, etc.)
- Users may override well known values if needed (e.g. some servers may require a specific user agent string)
- The Roku box will validate the data set in the headers to ensure it's of the right type, not too long, etc.

`Boolean SetHeaders(roAssociativeArrayString nameValueMap)`

- Add the specified HTTP headers in the nameValueMap associative array.
- Header limitations specified in AddHeader() still apply.

`Boolean SetCertificatesFile(String path)`

- Set the certificates file used for SSL to the .pem file specified. The .pem file should include the ca (certificate authority) cert that signed the cert installed on your web server. This enables authentication of your server.
- Instances of roUrlTransfer and ifHttpAgent components should call this function before performing https requests.
- The appropriate certificates file should be placed at the location specified in the SetCertificatesFile() function call.

`Boolean InitClientCertificates(Void)`

- Initialize the object to send the Roku client cert.

Example Code: ifHttpAgent

```
ba = CreateObject("roByteArray")
ba.FromAsciiString("myusername:mypassword")
myhttpagent.AddHeader("Authorization", "Basic " + ba.ToBase64String())
```

Interface: ifSetMessagePort

`Void SetMessagePort(roMessagePort port)`

- Set the message port to be used for all events from the screen.

Interface: ifGetMessagePort

`Object GetMessagePort(Void)`

- Returns the message port currently set for use by the screen.

Events: roPosterScreenEvent

The roPosterScreen sends the roPosterScreenEvent with the following predicates that indicate its valid event types:

`isScreenClosed()`

- Event type indicating that the screen was closed and no longer displayed to the user.
- Event Details: Type = 1, Msg = "", Index = 0, Data = 0.

`isListFocused()`

- Event type indicating that a new category in the filter banner has gained focus.
- Event Details: Type = 2, Msg = "", Index = Index of currently focused item, Data = 0.

`isListSelected()`

- Event type indicating that a category on the filter banner has been selected.

- Event Details: Type = 3, Msg = "", Index = Index of currently selected item, Data = 0.
- isListItemFocused()**
- Event type indicating that a new content item in the poster screen has gained focus.
 - Event Details: Type = 4, Msg = "", Index = Index of currently focused item, Data = 0.
- isListItemSelected()**
- Event type indicating that a content item in the poster screen has been selected.
 - Event Details: Type = 0, Msg = "", Index = Index of currently selected item, Data = 0.
- isAdSelected()**
- Event type indicating that the banner Ad in the poster screen has been selected.
 - Event Details: Type = 17, Msg = "", Index = 0, Data = 0.

Since Firmware version 2.7:

- isListItemInfo()**
- Event type indicating that the info button was pressed on a content item in the poster screen
 - Event Details: Type = 22, Msg = "", Index = Index of currently selected item, Data = 0.

Example Code: roPosterScreen

```
Function Main()
    port = CreateObject("roMessagePort")
    poster = CreateObject("roPosterScreen")
    poster.SetBreadcrumbText("[location1]", "[location2]")
    poster.SetMessagePort(port)

    list = CreateObject("roArray", 10, true)

    for i = 0 to 10
        o = CreateObject("roAssociativeArray")
        o.ContentType = "episode"
        o.Title = "[Title]"
        o.ShortDescriptionLine1 = "[ShortDescriptionLine1]"
        o.ShortDescriptionLine2 = "[ShortDescriptionLine2]"
        o.Description = ""
        o.Description = "[Description]"
        o.Rating = "NR"
        o.StarRating = "75"
        o.ReleaseDate = "[<mm/dd/yyyy]"
        o.Length = 5400
        o.Categories = []
        o.Categories.Push("[Category1]")
        o.Categories.Push("[Category2]")
        o.Categories.Push("[Category3]")
        o.actors = []
        o.actors.Push("[Actor1]")
        o.actors.Push("[Actor2]")
        o.actors.Push("[Actor3]")
        o.Director = "[Director]"
```

```

        list.Push(o)
    end for

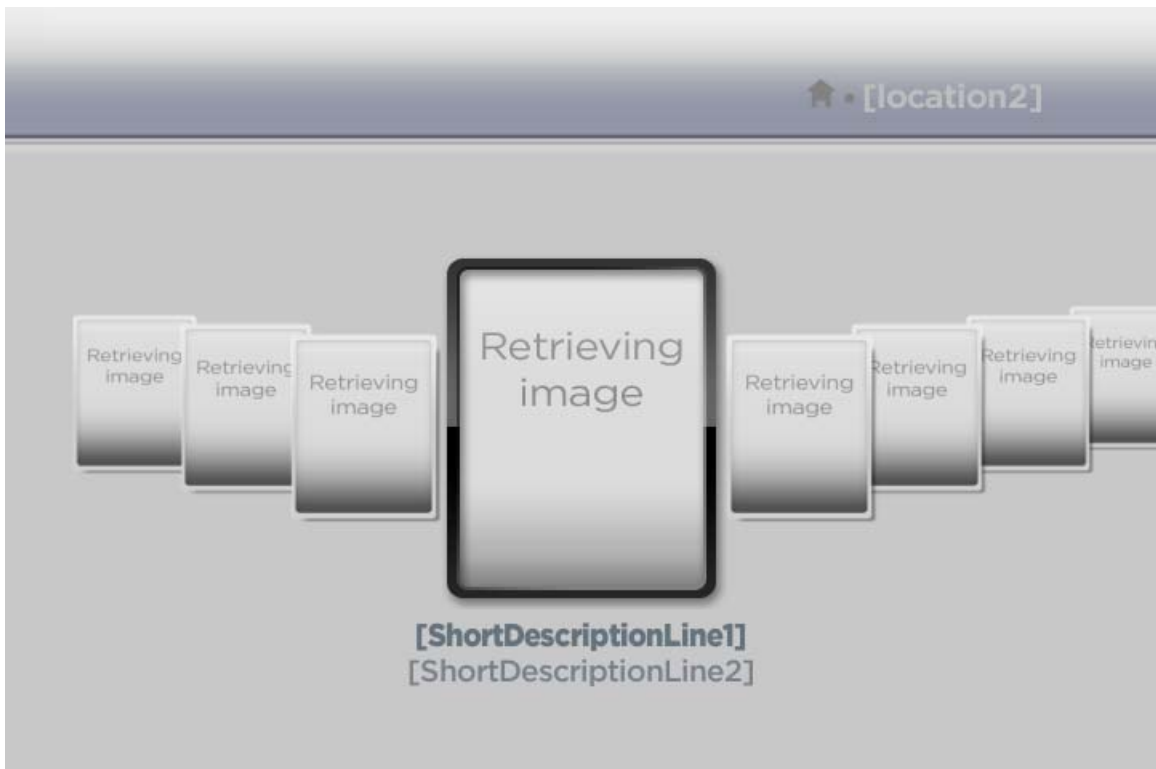
    poster.SetContentList(list)
    poster.Show()

    while true
        msg = wait(0, port)
        if msg.isScreenClosed() then
            return -1
        elseif msg.isButtonPressed()
            print "msg: ";msg.GetMessage();"idx: ";msg.GetIndex()
        endif
    end while
End Function

```

Image: roPosterScreen example results

The following screen is displayed when this code is executed:



Note: In this example, we did not set valid values for SDPosterUrl or HDPoster URL, so no artwork is displayed. We have also elected not to display the filter banner and did not call the SetListNames API, so no filter banner is displayed. The screen is fully functional in other respects and responds to user input, scrolls left/right and receives events as the poster selection changes.

4.4 roSpringboardScreen

The Springboard Screen shows detailed information about an individual piece of content and provides options for actions that may be taken on that content. The detailed description of the content is displayed with poster art for the title. Artwork may be displayed portrait or landscape orientation depending on the ContentType set in the metadata.

The caller may add one or more buttons to the screen with actions such as Play, Resume, Purchase or More Info. The script is notified via an event when a button is selected and it is the responsibility of the script writer to handle that event as desired and perform the requested action.

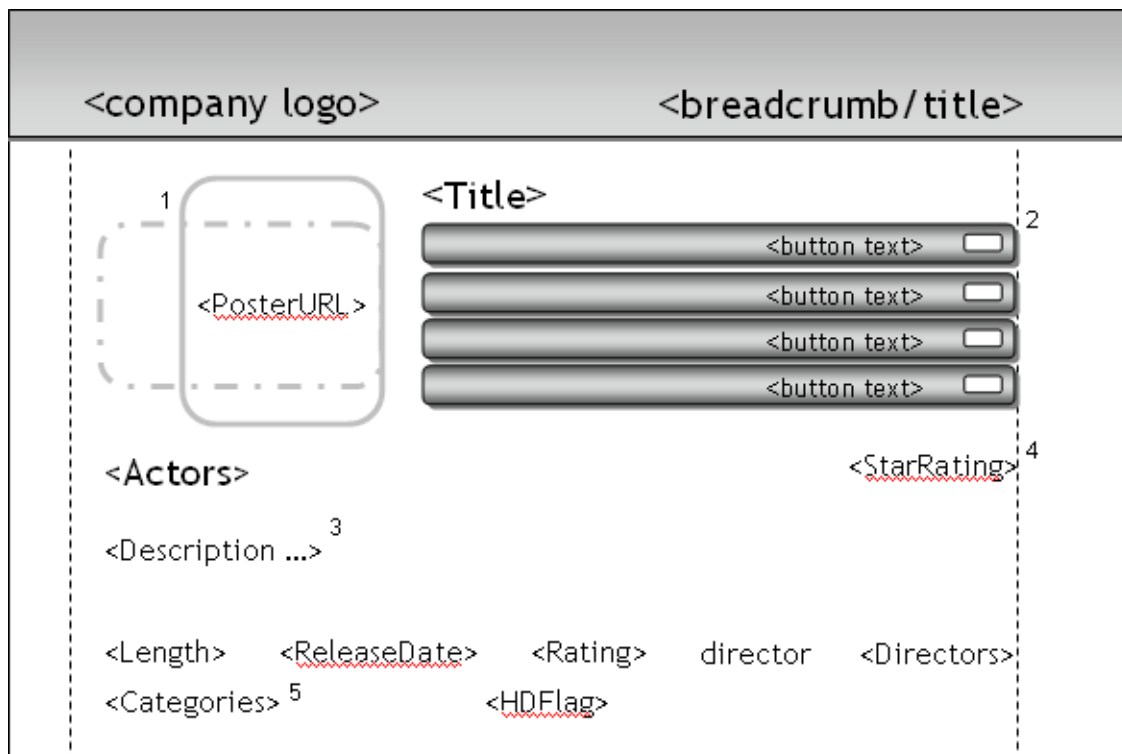


Diagram: roSpringboard video screen

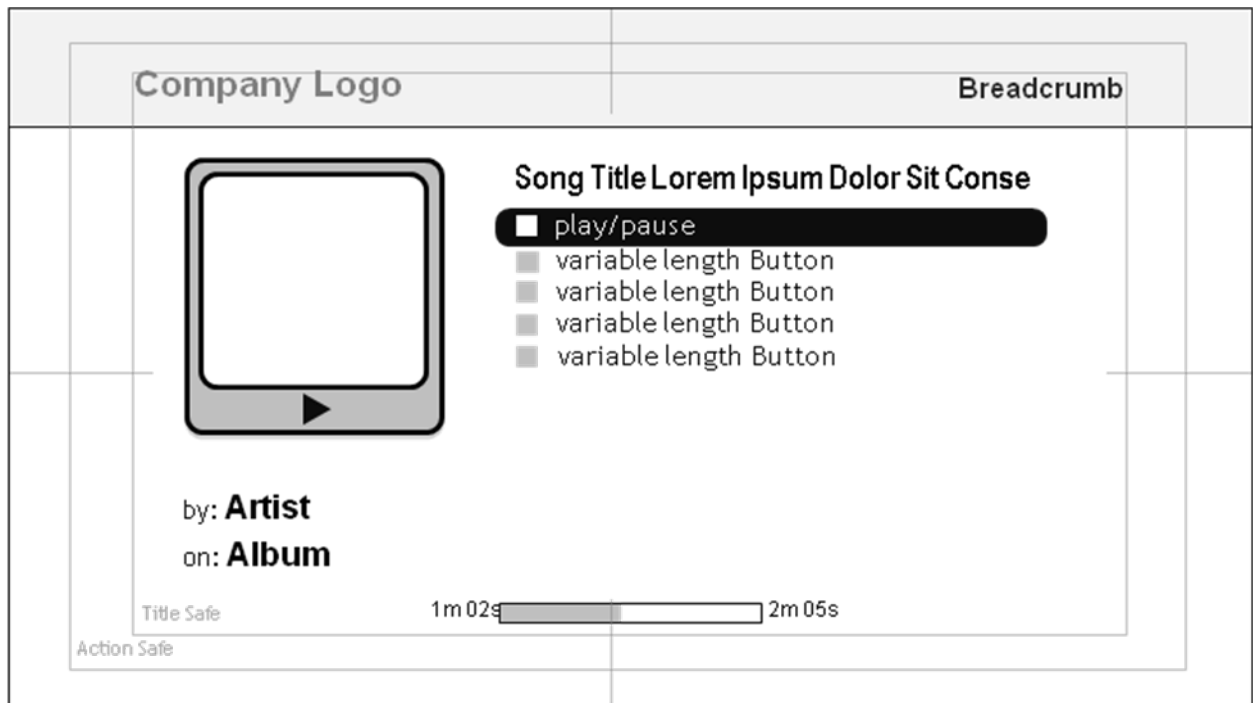


Diagram: roSpringboard audio screen, ContentType=audio

Notes:

1. Orientation for artwork is based on ContentType setting and may be portrait or landscape.
2. The audio springboard is capable of adding a progress bar.
3. If the ContentType is audio, the album art dimensions are:
SD: 124 x 112
HD: 188 x 188
4. If the ContentType is episode, the album art dimensions are:
SD: 180 x 122
HD: 264 x 198
5. If the ContentType is any other value, the album art dimensions are:
SD: 112 x 142
HD: 148 x 212
6. Up to 5 user-defined buttons may be displayed. Buttons are displayed in the order they are added and always appear in a fixed region of the screen
7. The description text will be formatted and justified (right and left edges) to fit between the margins. When the maximum length is reached, the text will be clipped and terminated with an ellipsis. The font is variable pitch, so the maximum number of characters is dependent on the text. The spacing is approximately 85 characters per line x 4 lines = 340 characters. The fonts and character spacing for HD and SD are similar, and display approximately the same number of characters, but the relationship is not exactly 1:1.
8. The star rating can show either community StarRating (red) or UserStarRating (yellow). If both values are set, the control will display the UserStarRating. If ratings are not desired, it can be removed by calling SetStaticRatingEnabled(false), providing more space to display actor names.

9. The Length attribute will display a formatted string or show length. If the value is zero, this field will display 0m, if the attribute is not set/missing then this field will not be displayed.

This object is created with no parameters:

- `CreateObject("roSpringboardScreen")`

The `roSpringboardScreen` implements the following interfaces:

Interface: `ifSpringboardScreen`

`Void SetContent(roAssociativeArray content)`

- Set the content to be displayed on the Springboard Screen. The content is passed by the caller as a content meta-data object describing the attributes for the title. See "Content Meta-Data" for details on the attributes for each title. Internally content meta-data is stored as an `roAssociativeArray`.
- Note that if the content type is audio (i.e, `content type == audio`), a progress indicator can be specified.

```

Void SetDescriptionStyle(String style)
    ▪ Set the springboard style for the poster screen. Styles allow the poster screen to look differently for different types of content or different usage. The following is the list of valid screen styles:
        ○ "audio": All tags on audio screen above are substituted with Content Meta-Data.
        ○ "movie": All tags on the video screen above are substituted with Content Meta-Data
        ○ "video": All tags except <Directors> and <Ratings> are on the video screen above are replaced with Content Meta-Data.
        ○ "generic": Only the <Title> and <Description> tags on the video screen above are replaced with Content Meta-Data.

Void SetProgressIndicatorEnabled(Boolean enable)
    ▪ Note that this will only work if the content type is audio (i.e, contentType == audio).

Void SetProgressIndicator(Integer progress, Integer max)
    ▪ progress ranges from [0 - N] and max ranges from [1 - N] (contentType == audio).
    ▪ Example:
        screen = CreateObject("roSpringboardScreen")
        screen.setProgressIndicatorEnabled(true)
        screen.show()
        'progress ranges from [0 - N] and max ranges from [1 - N]
        screen.setProgressIndicator(progress, max)

Boolean AddThumbsUpDownButton(Integer buttonID, Integer thumbRating)
    ▪ Adds a button to the screen identified buttonID provided. This is just a special type of button that is displayed in the standard location on the screen and appears in the order added sequenced with other buttons. The ID is defined by the developer and used to uniquely identify the button instance. When the button is pressed, the script will receive an event from the application indicating containing the ID of the button pressed and allowing the script to perform the desired action for that case. The thumbRating takes the values of -1, 0, or 1 and corresponds to thumbs down, no rating, and thumbs up.

Boolean AddThumbsUpDownButtonWithTips(Integer buttonID,
                                       Integer thumbRating, Array tipText)
    ▪ Behaves like addThumbsUpDownButton() and adds a tipText as an array of strings (size 2) which lets the script override the default tip text ["didn't like it", "liked it"]

Boolean AddButton(Integer id, String title)
    ▪ Adds a button to the screen identified by the title and ID provided. The buttons are displayed in a standard location on the screen and appear in the order added. The ID is defined by the developer and used to uniquely identify the button instance. When the button is pressed, the script will receive an event from the application containing the ID of the button pressed and allowing the script to perform the desired action for that case.

Boolean AddRatingButton(Integer id, Integer userRating,
                       Integer aggregateRating)
    ▪ Adds a star rating button to the screen. The rating is specified as a string whose value is an integer 1-100 which indicates the number of stars (1 to 5) to be displayed. Think of this as a percentage value <20% = 1 star. This button is displayed in the menu area of the screen and appears in the order added. The userRating specifies the value the user rated the title, while the aggregate Rating represents the total for all users. The userRating takes precedence and determines the color of the buttons if set. The aggregateRating may display half-stars. The button returns the userRating in the event data field.

Void SetStaticRatingEnabled(Boolean isEnabled)
    ▪ Sets the display of the static rating stars on or off. Defaults to enabled. When you want to use the ratings button or disable ratings on screen, set this value to false to remove this control from the screen. The extra space available will be used to display additional actors.

```

Void ClearButtons(Void)

- Clears all of the buttons from the screen and resets the array of buttons back to default with no buttons set.

Void SetBreadcrumbText(String location1, String location2)

- Breadcrumbs allow the application to display a two-part navigational title which shows the current and the previous locations in the application hierarchy (e.g. TV – Friends). If both location values are set, the application will display the title in breadcrumb format. If only the first location is set, the application will display the specified text in the title area like the SetTitle API call.

Void SetBreadcrumbEnabled(Boolean enable)

- Show or hide the breadcrumb text in the title area.

Void PrefetchPoster(String sdPosterURL, String hdPosterURL)

- Allows the screen to pre-fetch the poster images before the screen is displayed as a display optimization technique. This is useful when doing left-right navigation between springboard screens. The pre-fetch is done, loading the image cache and then the screen is displayed.

Void AllowUpdates(Boolean)

- When adding multiple buttons to the springboard dynamically, sometimes it is desirable to defer screen updates temporarily to avoid flashing. Call AllowUpdates(false), add several buttons and then call AllowUpdates(true) to refresh the display.

Integer CountButtons(Void)

- Returns the count of the number of buttons in the button menu on the Springboard.

Void AllowNavLeft(Boolean isAllowed)

- Enable or disable navigating left on the springboard screen.

Void AllowNavRight(Boolean isAllowed)

- Enable or disable navigating right on the springboard screen.

Void AllowNavRewind(Boolean isAllowed)

- Enable or disable sending the rewind remote event to script. Default is disabled.

Void AllowNavFastForward(Boolean isAllowed)

- Enable or disable sending the fast forward remote event to script. Default is disabled.

Void SetPosterStyle(String style)

- Set the display style for the poster screen. The default style is dependent on the content type of the poster, but this method enables the override of the screen type to one of the following:
 - “rounded-square-generic” – Artwork sizes: SD = 143x129; HD = 209x209 Note: is non-square NTSC pixel aspect ratio images. (Default)
 - “rounded-rect-16x9-generic” – 16x9 poster style. Artwork Sizes SD=177x90; HD=269x152

Since Firmware version 2.6:

- “multiple-portrait-generic” – Artwork sizes: SD = 104x134 ; HD =142x202

Void SetAdURL(String sdAdURL, String hdAdURL)

- Set the URL of the banner Ad image to be displayed on the springboard screen. This is currently only valid for the “video” style of springboard screen. Banner Ad sizes are as follows:
Banner Ad HD = 728x90 Banner Ad SD = 540x60

Void SetAdSelectable(Boolean isSelectable)

- Sets the banner ad to be selectable or display only. By default the banner ad is not selectable. When enabled the user can navigate and move focus to the banner image. When selected, the screen will receive an roPosterScreenEvent and the msg will return true for isAdSelected().

Void SetAdDisplayMode(String displayMode)

- Sets the scale mode for displaying ad images. Valid display modes are:
 - scale-to-fill – scale image to completely fill the rectangle of the bounding frame (default)
 - scale-to-fit. – scale image to fit horizontally or vertically as appropriate while still maintaining aspect ratio. This is the preferred display mode for ads.

`Boolean Show(Void)`

- Display or refresh the screen after initial creation or state changes.

`Void Close(Void)`

- Close the screen and delete the associated object. Useful for avoiding screen flicker when the display order of your screens does not resemble a stack.

`Void SetTitle(String title)`

- Set the title for the screen to the specified string.

Since Firmware version 2.6:

`Void UseStableFocus(Boolean enable)`

- When enable is true, keeps the highlighted button the same on subsequent calls to `Show()`
- When enable is false, resets the focused button to the top button on subsequent calls to `Show()`
- Default is false
- Generally, you will want to use the default value of false, because when navigating through springboards (using the left and right arrows) the user will want the top button to be highlighted.
- Can be useful to set to true when you wish to update other dynamic text on the same screenboard and keep the highlighted button static.

Since Firmware version 3.0:

`Void SetDisplayMode(String displayMode)`

- Sets the mode for displaying slideshow images. This allows images to be either scaled to completely fill the screen (scale-to-fill) or scaled to fit inside the screen (scale-to-fit) while maintaining aspect ratio. Valid display modes are:
 - scale-to-fill – scale image to completely fill the rectangle of the bounding frame (Default)
 - scale-to-fit. – scale image to fit horizontally or vertically as appropriate while still maintaining aspect ratio. Note that scale-to-fit may result in pillar-box or letter-box display of images.
 - zoom-to-fill – scales and crops image to maintain aspect ratio and completely fill the rectangle of the bounding frame.
 - photo-fit – Uses several methods to fit the image with a different aspect ratio to the screen. First, it will asymmetrically scale up to a maximum of 5%. Second, for landscape images, if vertical cropping is necessary, it will remove two lines off the bottom for every one line off the top up to a maximum of 30% of the image. For all images, if horizontal cropping is necessary it will crop an equal amount from both sides.

Interface: `ifHttpAgent`

The `ifHttpAgent` methods setup the way URLs in the Content Meta-Data are accessed via this object.

`Boolean AddHeader(String name, String value)`

- Add the specified HTTP header.

- If “x-roku-reserved-dev-id” is passed as name, the Roku box ignores the passed in value. In its place, sends the devid of the application publisher of the currently running app. This allows the developer’s server to know which client app is talking to it.
- Developers may set any headers except what’s explicitly reserved by Roku. The Roku box will enforce ownership of values in the namespace x-roku-reserved-
- The Roku box will set initial values of well known headers (e.g. User-Agent, Content-Length, etc.)
- Users may override well known values if needed (e.g. some servers may require a specific user agent string)
- The Roku box will validate the data set in the headers to ensure it’s of the right type, not too long, etc.

```
Boolean SetHeaders(roAssociativeArrayString nameValueMap)
```

- Add the specified HTTP headers in the nameValueMap associative array.
- Header limitations specified in AddHeader() still apply.

```
Boolean SetCertificatesFile(String path)
```

- Set the certificates file used for SSL to the .pem file specified. The .pem file should include the ca (certificate authority) cert that signed the cert installed on your web server. This enables authentication of your server.
- Instances of roUrlTransfer and ifHttpAgent components should call this function before performing https requests.
- The appropriate certificates file should be placed at the location specified in the SetCertificatesFile() function call.

```
Boolean InitClientCertificates(Void)
```

- Initialize the object to send the Roku client cert.

Example Code: ifHttpAgent

```
ba = CreateObject("roByteArray")
ba.FromAsciiString("myusername:mypassword")
myhttpagent.AddHeader("Authorization", "Basic " + ba.ToBase64String())
```

Interface: ifSetMessagePort

```
Void SetMessagePort(roMessagePort port)
```

- Set the message port to be used for all events from the screen.

Interface: ifGetMessagePort

```
Object GetMessagePort(Void)
```

- Returns the message port currently set for use by the screen.

Events: roSpringboardScreenEvent

The roSpringboardScreen sends the roSpringboardScreenEvent with the following predicates that indicate its valid event types:

```
isButtonPressed()
```

- Event indicating a button on the screen was selected and the ID of the button that was selected.
- Event Details: Type = 5, Msg = "", Index = ID of button from AddButton(), Data = 0 –or– value of user selection if the button type is a rating button.

```
isScreenClosed()
```

- Event indicating that the screen was closed and no longer displayed to the user.
 - Event Details: Type = 1, Msg = "", Index = 0, Data = 0
- isRemoteKeyPressed()**
- Event indicating the user pressed a key on their remote control. Currently only left/right navigation events are passed which allow the caller to navigate the content list from the springboard level.
 - Event Details: Type = 7, Msg = "", Index = Integer ID of key pressed, Data = 0

Example Code: roSpringboardScreen

The following example shows the process of creating an roSpringboardScreen, setting up the content meta-data, showing the screen and waiting for an event. This example is simplified for clarity and it's assumed the real-world applications will use techniques like getting data from web services using roUrlTransfer.

```
Function Main()
    port = CreateObject("roMessagePort")
    springBoard = CreateObject("roSpringboardScreen")
    springBoard.SetBreadcrumbText("[location 1]", "[location2]")
    springBoard.SetMessagePort(port)

    o = CreateObject("roAssociativeArray")
    o.ContentType = "episode"
    o.Title = "[Title]"
    o.ShortDescriptionLine1 = "[ShortDescriptionLine1]"
    o.ShortDescriptionLine2 = "[ShortDescriptionLine2]"
    o.Description = ""
    for i = 1 to 15
        o.Description = o.Description + "[Description] "
    end for
    o.SDPosterUrl = ""
    o.HDPosterUrl = ""
    o.Rating = "NR"
    o.StarRating = "75"
    o.ReleaseDate = "[mm/dd/yyyy]"
    o.Length = 5400
    o.Categories = CreateObject("roArray", 10, true)
    o.Categories.Push("[Category1]")
    o.Categories.Push("[Category2]")
    o.Categories.Push("[Category3]")
    o.actors = CreateObject("roArray", 10, true)
    o.actors.Push("[Actor1]")
    o.actors.Push("[Actor2]")
    o.actors.Push("[Actor3]")
    o.Director = "[Director]"

    springBoard.SetContent(o)
    springBoard.Show()

    while true
        msg = wait(0, port)
        if msg.isScreenClosed() then
            return -1
        elseif msg.isButtonPressed()
            print "msg: "; msg.GetMessage(); "idx: "; msg.GetIndex()
        endif
    end while
end Function
```

```
end while  
End Function
```

Image: roSpringboardScreen example results

The following screen is displayed when this code is executed:



4.5 roVideoScreen

The Video Screen object implements the video playback portion of the user interface. The API's to the video screen allow the developer to setup a fully featured playback environment with minimal coding. The developer is responsible for initial playback setup and providing the required data (e.g. StreamURLs, SteamsBitrates, etc.) as part of the content metadata. Once created and displayed, the screen will respond to events and manage the playback experience for the user.

The roVideoScreen is designed for streaming content. The preferred implementation should provide multiple bitrates (ideally four) of video to provide a high quality user experience under a variety of network conditions. Using the StreamBitrates and StreamURLs provided in the content meta-data for the item, the roVideoScreen will automatically monitor and select the best stream based on the users measured bandwidth. If network performance changes, the system will adapt and rebuffer to stream at a different bandwidth if necessary. Note that the StreamURLs, StreamBitrates, StreamQualities and StreamStickyHttpRedirects are all arrays that are aligned with each other. For example, the first stream listed would be the 0th element of all of these arrays.

The bitrates should represent the actual bitrate of the stream. The bitrate is used for both the display of the dots as well as the stream selection algorithm.

The dots work like this:

If the stream bitrate equals:

- 0 = no dots
- < 500 Kbps = 1 dot
- < 800 Kbps = 2 dots
- < 1.1 Mbps = 3 dots
- >= 1.1 Mbps = 4 dots

The dots are displayed automatically based on the bitrate of the stream selected unless there is a single stream and the bitrate is set to zero, then it won't show any dots. The StreamQuality attribute is used to select streams and indicates if a stream is HD or not. If the attribute for HDBranded is set to true and the stream is HD, the HD icon will show beside the quality dots. If the StreamQuality is set to HD, and the user display type is set to SD, the HD stream will not be selected.

The roVideoScreen automatically provides trick mode for all supported content types. There are two type of trick modes supported; scene based selection and time-based selection. If BIF image files are provided for a title, scene-based trick modes will appear. (See the BIF File format Documentation for more information) The user will be presented with the images and progress bar needed for SEEK, FF, REW within a stream. The following image shows how trick modes are rendered with BIF files:



Image: Trick Mode Support using BIF Images

The FF/REW functionality provides three speeds; slow, medium and fast. At slower speeds, the system displays the current frame in the center of the screen and additional frames on the side for contextual information. At higher speeds, the side frames disappear and only the center image is displayed. The I-frames within the video do not need to precisely align with the time stamp of the image frames in the BIF file. When the user stops and selects a frame, the video playback begins at the first I-frame less than or equal to the time position of the selected frame.

When BIF images are not available, the system will default to a time based trick play behavior. The user control is still the same, but only the progress bar is displayed and the user will not see individual scenes within the video. This mode is the default, so if images are not available for an individual title, the system will always provide this functionality by default.

The system will only seek to locations on an I-Frame boundary. Window Media (WMA9 or VC-1) uses the simple index object to determine the I-frame locations and H.264 uses the MOOV atom to determine the correct offsets. If the BIF images are at a consistent time intervals which do not align to I-Frame boundaries, the system will use the nearest I-Frame less than or equal to the time of the BIF image. MP4 or Windows Media are the preferred formats.

Note that BIF trick mode is not currently supported in the HLS Http Live Streaming player.

The following is a summary of supported video formats that have been tested and/or are currently in-use. Other formats or encoding may be supported, but should be evaluated on a case by case basis.

Table: Supported Video Encodings

| | H.264 SD | H.264 HD |
|--|---|---|
| Aspect Ratio ¹ | 4:3 | 16:9 |
| Dimension | Various to 720x480 | Various to 1280x720 and 1920x1080 for 1080p |
| Progressive/Interlaced | Progressive | Progressive |
| File Format | .mp4 (MPEG-4 Pt 14), .mov .m4v HLS: m3u8 & .ts | .mp4 (MPEG-4 Pt 14), .mov .m4v HLS: m3u8 & .ts |
| Frame Rate ² | 23.976 fps or 29.97 fps | 23.976 fps or 29.97 fps |
| Color Space | YUV | YUV |
| Video Codec | H.264/AVC | H.264/AVC |
| Profile | Main/High | High |
| Level/Complexity | 4.0 | 4.0 |
| Video Mode | Constrained VBR | Constrained VBR |
| Average Streaming Video Bitrate ³ | 384Kbps – 2.0Mbps | 1.6Mbps – 4.5Mbps |
| Average USB Video Bitrate ³ | 384Kbps – 8.0Mbps | 384Kbps – 8.0Mbps |
| Peak Video Bitrate | 1.5x average | 1.5x average |
| Key Frame Interval | < 10s | < 10s |
| DRM | None | None |
| Audio Codec | AAC LC AC3 Passthrough | AAC LC AC3 Passthrough |
| Audio Bit Rate | 128-256Kbps | 32-256Kbps |
| Audio Sample Rate | 44.1 Khz or 48Khz | 44.1 Khz or 48 Khz |
| Audio Sample Size | 16-Bit | 16-Bit |
| Audio Channels | 2-Ch Stereo | 2-Ch Stereo |

Notes: Important notes on video playback

- 1) The dimensions vary on a title-by-title basis depending on the source material and the target aspect ratio for the encode (e.g. 4:3 or 16:9). Content is always encoded at full width and the height is adjusted. For example, a 1.66 aspect ratio source is encoded as a 720x432 video and displayed as letterboxed for a 4:3 display.
- 2) The frame rate used for encoding is dependent on the source material. Film content is generally 23.976 fps, while video content is generally at 29.97.
- 3) For typical streaming video applications, we recommend a range of ~384Kbps to ~4.5Mbps. For USB playback, we recommend that you stay under 8.0 Mbps. This provides a good balance between quality and support for a wide number of users. In some cases lower and higher bitrates have been used, but this frequently results in poor quality or limits the % of the installed base that can view this encoding
- 4) It is critical that the StreamURLs, StreamBitrates, StreamQualities and StreamStickyHttpRedirects arrays are all aligned with each other. For example, the first stream listed would be the 0th element of all of these arrays. You may have multiple streams in the arrays and the system will automatically pick the most appropriate stream based on the users available bandwidth and video settings.

- 5) The StreamQualities array identifies streams as either SD or HD. If the user is configured for SD the system will not select HD streams for playback.
- 6) The optional StreamStartTimeOffset is the offset into the stream which is considered to be the new origin of playback.
- 7) Live – declares the video as live and replaces the time remaining text in the progress bar with “live”.
- 8) HLS Http Live Streaming support is included in the firmware (Introduced in firmware v2.6). We currently support version 3 of the Http Live Streaming protocol (Pantos – Draft submitted to IETF November 19, 2010 <http://tools.ietf.org/html/draft-pantos-http-live-streaming-05>). When using HLS, the StreamUrls and StreamQualities array should each have exactly one element. If the HLS stream has only a single bitrate stream, the StreamBitrates array should contain one element specifying that bitrate. If the stream contains more than one variant stream at multiple bitrates, the StreamBitrates array should contain one element with a value of zero. Please see the Video Encoding Guide for information about creating HLS .m3u8 files and segmented .ts files from your current h264 encoded video or distributing live video over HLS to the Roku box.
- 9) In addition to the support for version 2 of the HLS Pantos draft spec, the Roku box supports .m3u8 files that are compressed via deflate or gzip. The HTTP response for a query that returns a gzip-compressed file must contain the header:

Content-Encoding: gzip

The HTTP response for a query that returns a deflate-compressed file must contain the header:

Content-Encoding: deflate

- 10) “Trick Modes” and seeking work a little differently with HLS streams. First, there is currently no BIF file support on HLS streams and the trick modes that BIF files support do not exist in HLS. Second, there are a couple of ways that seeking works with HLS and they are different than other streams.

One way of seeking uses the “target duration” specified in the .m3u8 file. The first segment in an m3u8 file is assigned a time offset:

$$T = G * N$$

where G is the “target duration” value and N is the sequence number of the segment. Each subsequent segment is assigned a time offset equal to T (the time offset of the first segment) plus the duration value of all earlier segments. The duration of a segment is determined by the EXTINF line before that segment.

For example, in this file:

```
#EXT-X-TARGETDURATION:10
#EXT-X-MEDIA-SEQUENCE:37
#EXTINF:10
url1
#EXTINF:8
url2
#EXTINF:10
url3
```

The segment url1 has a time offset of 370, url2 is 380, and url3 is 388. Note that if no TARGETDURATION is specified, the default is 1, so the first segment in the file will have a nonzero time offset (equal to the target duration). The PlayStart content-meta data value allows direct seeking to an offset that is valid within the window of data in the current .m3u8 file.

There is a second way to seek in an HLS stream. If the m3u8 file has #EXT-X-PROGRAM-DATE-TIME entries, you can seek to a particular date/time by

passing a value equal to a modified Unix epoch value. The modified epoch is 1/1/2004 rather than the standard Unix epoch of 1/1/1970. A Unix time value can be converted to an HLS seek time by subtracting 1072915200 (the number of seconds between 1/1/1970 and 1/1/2004). Once again, setting the PlayStart content meta data value allows direct seeking to a specific time offset.

For example, to seek to the segment marked with the date/time of 7/4/2010 11:30, set PlayStart to 205327800. An example shell expression showing this arithmetic is:

```
% expr `date -d "7/4/2010T11:30:00.000" +%s` - 1072915200
205327800
```

In BrightScript, the same calculation might be:

```
dt = CreateObject("roDateTime")
dt.fromISO8601String("7/4/2010T11:30:00.000")
itemContentMetaData.PlayStart = dt.asSeconds() - 1072915200 '205327800
```

- 11) In firmware version 2.6, we've introduced support for SRT files. Please see the content meta-data parameter SubtitleUrl for pointing to a matching SRT file for your video content.
- 12) In firmware version 2.7, we've introduced 1080p support. Please see the content meta-data parameter FullHD for specifying 1080p resolution. Playback at 1080p resolution will only occur when the user has set the display type to HDTV 1080p. Another content meta-data parameter, FrameRate, specifies the frames per second of the video. Valid values are 24 and 30. If the user's display type is set to 1080p and FullHD for the content is false or not set, HD playback will be at 720p resolution. If the user's display type is set to HDTV 720p and FullHD content is set to 1080p resolution, the box will downscale the content to 720p resolution.

This object is created with no parameters:

```
CreateObject("roVideoScreen")
```

The roVideoScreen implements the following interfaces:

Interface: ifVideoScreen

```
Void SetContent(roAssociativeArray content)
    ▪ Set the content to be played in the roVideoScreen. The content is passed by the caller as an roAssociativeArray describing the attributes for the title. See "Content Meta-Data" for details on the attributes for each element in the array.
Void SetPositionNotificationPeriod(Integer period)
    ▪ Set interval to receive playback position events from the roVideoScreen. The notification period is specified in seconds. Notification events sent to the script specify the position in seconds relative to the beginning of the stream.
Void SetPreviewMode(Boolean enable)
    ▪ Set preview mode on/off. The preview mode allows the playback of a fixed duration (2 minute) preview of the movie.
Void SetCGMS(Integer level)
    ▪ Set CGMS (Copy Guard Management System) on analog outputs to the desired level.
        ○ 0 - CGMS off
        ○ 1 - No Copy Restriction
        ○ 2 - Copy No More
        ○ 3 - Copy Once Allowed
        ○ 4 - No Copying Permitted
Boolean Show(Void)
    ▪ Display or refresh the screen after creation or state changes.
Void Close(Void)
```


- Close the screen and delete the associated object. Useful for avoiding screen flicker when the display order of your screens does not resemble a stack.

Since Firmware version 2.6:

`Void Pause(Void)`

- Programatically pause the video.
- Behaves as if the user hit the pause button.

`Void Resume(Void)`

- Programatically resume the video.
- Behaves as if the user resumed the video.

`Void Seek(Integer milliseconds)`

- Set the play start offset to the specified milliseconds.

`Void SetDestinationRect(roAssociativeArray rect)`

- Set the target display window for the video.
- rect has the params: {x: Integer, y: Integer, w: Integer, h: Integer}
- Default value is: {x:0, y:0, w:0, h:0}, which is full screen
- Only useful if you'd like to zoom your video content (maybe to remove letterboxing) and you know the pixel offsets. For example, you may know that your 4:3 content has letterboxing for 16:9 video and your app is trying to display on an HD 16:9 display. In this case a rect value of {-135,-132, w:1440, h:1136} may crop out the letter boxing and zoom the video content to fit the screen. (Actual rect values are content dependent).

Since Firmware version 3.0:

`Void SetMaxVideoDecodeResolution(Integer width, Integer height)`

- Set the max resolution required by your video.
- Video decode memory is a shared resource with OpenGL texture memory. The Brightscript 2D APIs are implemented using OpenGL texture memory on Roku models that support the Open GL APIs (please see Section 1.4 of the Developer Guide for a list of these models).
- On models that do not support Open GL APIs this method exists for API compatibility but has no effect on actual memory allocations.
- Video decode memory allocation is based on a resolution of 1920x1080 as the maximum supported resolution.
- This API enables applications that want to use both the 2D APIs and video playback with a lower resolution than 1080p. Without this call, these applications are likely to not have enough memory for either video playback or roScreen rendering.

Deprecated Since Firmware version 3.0:

`Void SetMacrovisionLevel(Integer level)`

- Deprecated stub function does nothing. Roku no longer supports Macrovision and this function exists as a no-op so that legacy scripts do not break.

Interface: ifHttpAgent

The ifHttpAgent methods setup the way URLs in the Content Meta-Data are accessed via this object.

`Boolean AddHeader(String name, String value)`

- Add the specified HTTP header.
- If “x-roku-reserved-dev-id” is passed as name, the Roku box ignores the passed in value. In its place, sends the devid of the application publisher of the currently running app. This allows the developer’s server to know which client app is talking to it.
- Developers may set any headers except what’s explicitly reserved by Roku. The Roku box will enforce ownership of values in the namespace x-roku-reserved-
- The Roku box will set initial values of well known headers (e.g. User-Agent, Content-Length, etc.)
- Users may override well known values if needed (e.g. some servers may require a specific user agent string)
- The Roku box will validate the data set in the headers to ensure it’s of the right type, not too long, etc.

Boolean SetHeaders(roAssociativeArrayString nameValueMap)

- Add the specified HTTP headers in the nameValueMap associative array.
- Header limitations specified in AddHeader() still apply.

Boolean SetCertificatesFile(String path)

- Set the certificates file used for SSL to the .pem file specified. The .pem file should include the ca (certificate authority) cert that signed the cert installed on your web server. This enables authentication of your server.
- Instances of roUrlTransfer and ifHttpAgent components should call this function before performing https requests.
- The appropriate certificates file should be placed at the location specified in the SetCertificatesFile() function call.

Boolean InitClientCertificates(Void)

- Initialize the object to send the Roku client cert.

Example Code: ifHttpAgent

```
ba = CreateObject("roByteArray")
ba.FromAsciiString("myusername:mypassword")
myhttpagent.AddHeader("Authorization", "Basic " + ba.ToBase64String())
```

Interface: ifSetMessagePort

Void SetMessagePort(roMessagePort port)

- Set the message port to be used for all events from the screen.

Interface: ifGetMessagePort

Object GetMessagePort(Void)

- Returns the message port currently set for use by the screen.

Events: roVideoScreenEvent

The roVideoScreen sends the roVideoScreenEvent with the following predicates that indicate its valid event types:

isScreenClosed()

- Event indicating that the screen was closed and no longer displayed to the user.
- Event Details: Type = 1, Msg = "", Index = 0, Data = 0

isPlaybackPosition()

- Periodic event to the script to indicate the current position in the video stream.

- Event Details: Type = 6, Msg = "", Index = position offset in seconds, Data = 0
- isRequestFailed()**
- Event to the script to indicate error in video playback
 - Event Details: Type = 9, Msg = <text description of error>, Index = <error id>, Data = <roku internal error>
 - Error number codes:
 - 0 = Network error : server down or unresponsive, server is unreachable, network setup problem on the client.
 - -1 = HTTP error: malformed headers or HTTP error result.
 - -2 = Connection timed out
 - -3 = Unknown error
 - Roku Internal Errors
 - When you get an error code of -3, the Roku Internal Error number may have further meaning to Roku Developer Support.
 - It's good to record these roku internal errors whenever you get a -3 Error number.

Since Firmware version 2.6:

- Info = { url:<url of video>,
StreamBitrate:<bitrate of stream>,
MeasuredBitrate:<actualBitrateOfDataTransfer>
}
- Info was Introduced in firmware v2.6

- isStatusMessage()**
- Event to script with status messages about playback.
 - Event Details: Type = 11, Msg = <video player defined messages>, Index = "", Index=0, Data = 0
- isFullResult()**
- Event to indicate that playback completed at end of content
 - Event Details: Event Details: Type = 16, Msg = "", Index = 0, Data = 0
- isPartialResult()**
- Event to indicate playback was interrupted
 - Event Details: Type = 15, Msg = "", Index = 0, Data = 0

Since Firmware version 2.6:

- isStreamStarted()**
- Event to give details about the stream that was started.
 - Event Details: Type = 20, Msg = "", Index = Seconds from Play to Start Streaming, Data = 0,
 - Info = { url:<url of video>, StreamBitrate:<bitrate of stream in kbps>,
MeasuredBitrate:<actualBitrateOfDataTransfer in kbps>,
IsUnderrun:<true if streamStarted is the result of an underrun> }
- isPaused()**
- Event type indicating that video playback was paused by the user.
 - Event Details: Type = 12, Msg = "", Index = 0, Data = 0
- isResumed()**
- Event type indicating that video playback has resumed.
 - Event Details: Type = 13, Msg = "", Index = 0, Data = 0

Example Code: roVideoScreen

```
*****
' This example function is passed an associative array representing a
' piece of content (e.g. a TV episode) There are other attributes
' (title, description, etc.) but this example focuses on showing
' attributes required for initiating playback. It creates a video
' screen, sets the content and starts playback by calling Show()
*****
Function showVideoScreen(episode As Object)
    if type(episode) <> "roAssociativeArray" then
        print "invalid data passed to showVideoScreen"
        return -1
    endif

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

    ' Note: HDBranded controls whether the "HD" logo is displayed for a
    ' title. This is separate from IsHD because its possible to
    ' have an HD title where you don't want to show the HD logo
    ' branding for the title. Set these two as appropriate for
    ' your content
    episode.HDBranded = false
    episode.IsHD = false

    ' Note: The preferred way to specify stream info in v2.6 is to use
    ' the Stream roAssociativeArray content meta data parameter.

    episode.Stream = { url:"http://myserver.mydomain.com/mycontent.mp4",
                        bitrate:2000
                        quality:false
                        contentid:"mycontent-2000"
                      }
    episode.StreamFormat: "mp4"

    ' now just tell the screen about the title to be played, set the
    ' message port for where you will receive events and call show to
    ' begin playback. You should see a buffering screen and then
    ' playback will start immediately when we have enough data buffered.
    screen.SetContent(episode)
    screen.SetMessagePort(port)
    screen.Show()

    ' Wait in a loop on the message port for events to be received.
    ' We will just quit the loop and return to the calling function
    ' when the users terminates playback, but there are other things
    ' you could do here like monitor playback position and see events
    ' from the streaming player. Look for status messages from the
video
    ' player for status and failure events that occur during playback

    while true
```

```

msg = wait(0, port)

if type(msg) = "roVideoScreenEvent" then
    print "showVideoScreen | msg = "; msg.GetMessage() " | index
= "; msg.GetIndex()
    if msg.isScreenClosed()
        print "Screen closed"
        exit while
    else if msg.isStatusMessage()
        print "status message: "; msg.GetMessage()
    else if msg.isPlaybackPosition()
        print "playback position: "; msg.GetIndex()
    else if msg.isFullResult()
        print "playback completed"
        exit while
    else if msg.isPartialResult()
        print "playback interrupted"
        exit while
    else if msg.isRequestFailed()
        print "request failed - error: "; msg.GetIndex(); " -
"; msg.GetMessage()
        exit while
    end if
end if
end while

End Function

```

4.6 roSlideShow

The Slide Show screen provides the ability to setup a photo slide show to playback a series of images. Images may be jpg, png or gif files. The developer can control the sequencing and timing of the slideshow. The object is designed to accept an array of content meta-data objects, describing the images and providing url's for accessing each image. There are content meta-data properties used to display a text overlay. They are TextOverlayUL, TextOverlayUR, and TextOverlayBody and are described in Section 3.3.

This object is created with no parameters:

- CreateObject("roSlideShow")

The roSlideShow implements the following interfaces:

Interface: ifSlideShow

```

Void SetContentList(Array contentList)
    ▪ Set the content to be played by the slide show. The caller passes an roArray of
    roAssociativeArrays (Content Meta-Data objects) representing the information for each
    title to be displayed on screen. See "Content Meta-Data" for details on the attributes for
    each element in the array.
Void AddContent(roAssociativeArray contentItem)
    ▪ Add a new ContentMetaData item to the content list for the slide show. New items are
    added to the end of the list.
Void ClearContent(Void)
    ▪ Clear all content from the SlideShow screen.
Integer CountButtons(Void)

```

- Returns the count of all buttons added to the slide show screen

```
Void SetNext(Integer index, Boolean isImmediate)
```

- Tells the SlideShow object to queue a particular slide up as the next slide. If immediate is true it forces an immediate update.

```
Void SetPeriod(Integer seconds)
```

- Defines the number of seconds that each slide is displayed.

```
Void SetTextOverlayHoldTime(Integer milliSeconds)
```

- Defines the number of milliseconds to display the text overlay for each slide. If set to zero, the overlay is off.

```
Boolean Pause(Void)
```

- Put the slide show into *pause* mode.
- It is an error to Pause if player is not in play mode.

```
Boolean Resume(Void)
```

- Put slide show into *play* mode starting from the pause point.
- It is an error to Resume from any other mode than Pause.

```
Void SetTextOverlayIsVisible(Boolean isVisible)
```

- If isVisible is true, display the overlay. If isVisible is false do not display the overlay. Note that it is OR'd with the overlay hold time. So even if isVisible is false, during the slide's overlay hold time the overlay is displayed.

```
Boolean AddButton(Integer id, String title)
```

- Adds a button to the screen identified by the title and ID provided. The buttons are displayed in a standard location on the screen and appear in the order added. The ID is defined by the developer and used to uniquely identify the button instance. When the button is pressed, the script will receive an event from the application containing the ID of the button pressed.

```
Boolean AddRatingButton(Integer id, Integer rating)
```

- Adds a star rating button to the screen. The rating is specified as a String that's value is an integer 1-100 which indicates the number of stars (1 to 5) to be displayed. Think of this as a percentage value <20% = 1 star, This button is displayed in a standard location on the screen and appear in the order added.

```
Void ClearButtons(Void)
```

- Clears all of the buttons from the screen and resets the array of buttons back to default with no buttons set.

```
Void SetUnderscan(Float percentage)
```

- Set the percentage to reduce the image size by to compensate for monitor overscan. E.g. 2.5 for 2.5%

```
Void SetDisplayMode(String displayMode)
```

- Sets the mode for displaying slideshow images. This allows images to be either scaled to completely fill the screen (scale-to-fill) or scaled to fit inside the screen (scale-to-fit) while maintaining aspect ratio. Valid display modes are:
 - scale-to-fill – scale image to completely fill the rectangle of the bounding frame (Default)
 - scale-to-fit. – scale image to fit horizontally or vertically as appropriate while still maintaining aspect ratio. Note that scale-to-fit may result in pillar-box or letter-box display of images.
 - zoom-to-fill – scales and crops image to maintain aspect ratio and completely fill the rectangle of the bounding frame.
 - photo-fit – Uses several methods to fit the image with a different aspect ratio to the screen. First, it will asymmetrically scale up to a maximum of 5%. Second, for landscape images, if vertical cropping is necessary, it will remove two lines off the bottom for every one line off the top up to a maximum of 30% of the image. For all images, if horizontal cropping is necessary it will crop an equal amount from both sides.

```
Void SetMaxUpscale(Float maxUpscale)
```

- Set the maximum scale factor for scale-to-fill, zoom-to-fill, and photo-fit modes.

Boolean Show(Void)

- Display or refresh the screen after creation or state changes.

Void Close(Void)

- Close the screen and delete the associated object. Useful for avoiding screen flicker when the display order of your screens does not resemble a stack.

Since Firmware version 3.0:

Void SetBorderColor(String color)

- Set the border color used as background around slide to passed color string containing the HTML Hex color value. ed by your video.
-

Interface: ifHttpAgent

The ifHttpAgent methods setup the way URLs in the Content Meta-Data are accessed via this object.

Boolean AddHeader(String name, String value)

- Add the specified HTTP header.
- If "x-roku-reserved-dev-id" is passed as name, the Roku box ignores the passed in value. In its place, sends the devid of the application publisher of the currently running app. This allows the developer's server to know which client app is talking to it.
- Developers may set any headers except what's explicitly reserved by Roku. The Roku box will enforce ownership of values in the namespace x-roku-reserved-
- The Roku box will set initial values of well known headers (e.g. User-Agent, Content-Length, etc.)
- Users may override well known values if needed (e.g. some servers may require a specific user agent string)
- The Roku box will validate the data set in the headers to ensure it's of the right type, not too long, etc.

Boolean SetHeaders(roAssociativeArrayString nameValueMap)

- Add the specified HTTP headers in the nameValueMap associative array.
- Header limitations specified in AddHeader() still apply.

Boolean SetCertificatesFile(String path)

- Set the certificates file used for SSL to the .pem file specified. The .pem file should include the ca (certificate authority) cert that signed the cert installed on your web server. This enables authentication of your server.
- Instances of roUrlTransfer and ifHttpAgent components should call this function before performing https requests.
- The appropriate certificates file should be placed at the location specified in the SetCertificatesFile() function call.

Boolean InitClientCertificates(Void)

- Initialize the object to send the Roku client cert.

Example Code: ifHttpAgent

```
ba = CreateObject("roByteArray")
ba.FromAsciiString("myusername:mypassword")
myhttpagent.AddHeader("Authorization", "Basic " + ba.ToBase64String())
```

Interface: ifSetMessagePort

Void SetMessagePort(roMessagePort port)

- Set the message port to be used for all events from the screen.

Interface: `ifGetMessagePort`

`Object GetMessagePort(Void)`

- Returns the `roMessagePort` currently set for use by the screen.

Events: `roSlideShowEvent`

The `roSlideShow` sends the `roSlideShowEvent` with the following predicates that indicate its valid event types:

- `isButtonPressed()`**
 - Event type indicating a button on the screen was selected.
 - Event Details: Type = 5, Msg = "", Index = ID of button from `AddButton()`, Data = 0
- `isScreenClosed()`**
 - Event type indicating that the screen was closed and no longer displayed to the user.
 - Event Details: Type = 1, Msg = "", Index = 0, Data = 0
- `isPlaybackPosition()`**
 - Event received on change of image in slide show
 - Event Details: Type = 6, Msg = "", Index = <Index of current image>, Data = 0
- `isRemoteKeyPressed()`**
 - Event type indicating that a IR remote key was pressed
 - Event Details: Type = 7, Msg = "", Index = <IR key as an Integer>, Data = 0
- `isRequestSucceeded()`**
 - Event to the script to indicate success in slideshow playback
 - Event Details: Type = 8, Msg = "", Index = <Index of decoded image>, Data = 0
- `isRequestFailed()`**
 - Event to the script to indicate error in slideshow playback
 - Event Details: Type = 9, Msg = "", Index = <Index of failed image>, Data = 0
- `isRequestInterrupted()`**
 - Event type indicating that the current image was interrupted
 - Event Details: Type = 10, Msg = "", Index = <Index of current image>, Data = 0
- `isPaused()`**
 - Event type indicating that the slide show was paused
 - Event Details: Type = 12, Msg = "", Index = 0, Data = 0
- `isResumed()`**
 - Event type indicating that slide show playback has resumed
 - Event Details: Type = 13, Msg = "", Index = 0, Data = 0

Example Code: `roSlideShow`

4.7 `roSearchScreen`

The Search Screen provides a standard way to allow users to enter text for searching. This screen features a simplified keyboard (a-z, 0-9) designed to provide just the keys necessary to perform case-insensitive searches without punctuation.

Ideally, the user would enter a search string and the backend service would perform that query in a case-insensitive manner ignoring special characters like punctuation. The script is notified as each key is pressed so that a progress disclosure search can be performed if supported by the

back-end service. In addition, the script can control the text displayed on the screen and will receive events when the text entry is complete.

In addition to entering search strings, this screen features a list that can be used to display search results or show the most recent searches. It's desirable for the screen to maintain a list of recent searches for the user to allow them to easily repeat a recent query without typing. In some implementations, it may be desirable to use this list to show a progressive set of results after each character while the user is typing.

This object is created with no parameters:

```
CreateObject("roSearchScreen")
```

The roSearchScreen implements the following interfaces:

Interface: ifSearchScreen

```
Void SetSearchTermHeaderText(String text)
    ▪ Set the text to be displayed for the header in the list area. This area could contain a list of
      search terms previously used as a search history or partial results in the case of a
      progressive disclosure search.
Void SetSearchButtonText(String text)
    ▪ Set the text for the search button at the bottom of the keyboard to the desired text string
Void SetClearButtonText(String text)
    ▪ Set the text label for the button at the bottom of the list area. Example text might be "clear
      history", "clear results" or similar.
Void AddSearchTerm(String searchTerm)
    ▪ Add an individual value to the search term list.
Void SetSearchTerms(roArray searchTerms)
    ▪ Set the search terms list to the values contained in the array provided. The argument is
      an array of string values to be displayed.
Void ClearSearchTerms(Void)
    ▪ Clear all values from the search terms list.
Void SetSearchButtonText(String text)
    ▪ Set the text label to be displayed on the search button. For example "search", "find", etc.
Void SetBreadcrumbText(String location1,
                       String location2)
    ▪ Breadcrumbs allow the application to display a two-part navigational title which shows the
      current and the previous locations in the application hierarchy (e.g. TV – Friends). If both
      location values are set, the application will display the title in breadcrumb format. If only
      the first location is set, the application will display the specified text in the title area like
      the setTitle API call.
Void SetBreadcrumbEnabled(Boolean enable)
    ▪ Show or hide the breadcrumb text in the title area.
Void SetClearButtonEnabled(Boolean enable)
    ▪ Show or hide the clear button on the keypad.
Void SetEmptySearchTermsText(String text)
    ▪ When there are no search terms, display the passed text param in the search terms box.
Void SetSearchText(String text)
    ▪ Set the keyboard search string box to the passed text.
Boolean Show(Void)
    ▪ Display or refresh the screen after creation or state changes.
Void Close(Void)
    ▪ Close the screen and delete the associated object. Useful for avoiding screen flicker
      when the display order of your screens does not resemble a stack.
```

Interface: ifSetMessagePort

```
Void SetMessagePort(roMessagePort port)
    ▪ Set the message port to be used for all events from the screen.
```

Interface: ifGetMessagePort

```
Object GetMessagePort(Void)
    ▪ Returns the message port currently set for use by the screen.
```

The roSearchScreen sends the roSearchScreenEvent with the following predicates that indicate its valid event types:

```
isScreenClosed()
    o Event indicating that the screen was closed and no longer displayed to the user.
    o Event Details: Type = 1, Msg = "", Index = 0, Data = 0
isCleared()
    o Event indicating that the search list has been cleared
    o Event Details: Type = 14, Msg = "", Index = 0, Data = 0
isPartialResult()
    o Event indicating that the results received are a partial entry based on last key
    o Event Details: Type = 15, Msg = <text of partial result>, Index = 0, Data = 0
isFullResult()
    o Event indicating that the results received are a complete search request
    o Event Details: Type = 16, Msg = <text of full result>, Index = 0, Data = 0
```

Example Code: roSearchScreen

```
REM *****
REM Main routine - example of search screen usage
REM *****

Sub Main()
    print "start"

    'toggle the search suggestions vs. search history behavior
    'this allow you to generate both versions of the example below
    displayHistory = false

    history = CreateObject("roArray", 1, true)
    'prepopulate the search history with sample results
    history.Push("seinfeld")
    history.Push("fraiser")
    history.Push("cheers")

    port = CreateObject("roMessagePort")
    screen = CreateObject("roSearchScreen")
    'commenting out SetBreadcrumbText() hides breadcrumb on screen
    screen.SetBreadcrumbText("", "search")
    screen.SetMessagePort(port)

    if displayHistory
        screen.SetSearchTermHeaderText("Recent Searches:")
        screen.SetSearchButtonText("search")
        screen.SetClearButtonText("clear history")
```

```

        screen.SetClearButtonEnabled(true) 'defaults to true
        screen.SetSearchTerms(history)
    else
        screen.SetSearchTermHeaderText("Suggestions:")
        screen.SetSearchButtonText("search")
        screen.SetClearButtonEnabled(false)
    endif

    print "Doing show screen..."
    screen.Show()

    print "Waiting for a message from the screen..."

    ` search screen main event loop
    done = false
    while done = false
        msg = wait(0, screen.GetMessagePort())

        if type(msg) = "roSearchScreenEvent"
            if msg.isScreenClosed()
                print "screen closed"
                done = true
            else if msg.isCleared()
                print "search terms cleared"
                history.Clear()
            else if msg.isPartialResult()
                print "partial search: "; msg.GetMessage()
                if not displayHistory
                    screen.SetSearchTerms((msg.GetMessage()))
                endif
            else if msg.isFullResult()
                print "full search: "; msg.GetMessage()
                history.Push(msg.GetMessage())
                if displayHistory
                    screen.AddSearchTerm(msg.GetMessage())
                end if
                'uncomment to exit the screen after a full search
            result:
                'done = true
            else
                print "Unknown event: "; msg.GetType(); " msg: ";sg.GetMessage()
            endif
        endif
    endwhile

    print "Exiting..."
End Sub

Function GenerateSearchSuggestions(partSearchText As String) As Object
    suggestions = CreateObject("roArray", 1, true)

    length = len(partSearchText)
    if length > 6
        suggestions.Push("ghost in the shell")
        suggestions.Push("parasite dolls")
        suggestions.Push("final fantasy")
    end if
end function

```

```

        suggestions.Push("ninja scroll")
        suggestions.Push("space ghost")
        suggestions.Push("hellboy")
    else if length > 5
        suggestions.Push("parasite dolls")
        suggestions.Push("final fantasy")
        suggestions.Push("ninja scroll")
        suggestions.Push("space ghost")
        suggestions.Push("hellboy")
    else if length > 4
        suggestions.Push("final fantasy")
        suggestions.Push("ninja scroll")
        suggestions.Push("space ghost")
        suggestions.Push("hellboy")
    else if length > 3
        suggestions.Push("ninja scroll")
        suggestions.Push("space ghost")
        suggestions.Push("hellboy")
    else if length > 2
        suggestions.Push("space ghost")
        suggestions.Push("hellboy")
    else if length > 1
        suggestions.Push("hellboy")
    else if length > 0
        suggestions.Push("transformers")
    endif

    return suggestions
End Function

```

Image: roSearchScreen example results (search suggestions)



Image: roSearchScreen example results (search history)

4.8 roSearchHistory

The Search History object implements the system wide storage of search terms for use in implementing the roSearchScreen. As the user searches for content, recent searches are placed into the roSearch history. This allows the user to easily re-execute these commands later without typing on the keyboard. The initial list of recent searches is displayed on the roSearchScreen to assist the user in finding content to watch. This history is used system wide, so that the user can find references to their search in multiple types of content.

This object is created with no parameters:

- `CreateObject("roSearchHistory")`

The roSearchHistory component implements the following interfaces:

Interface: ifSearchHistory

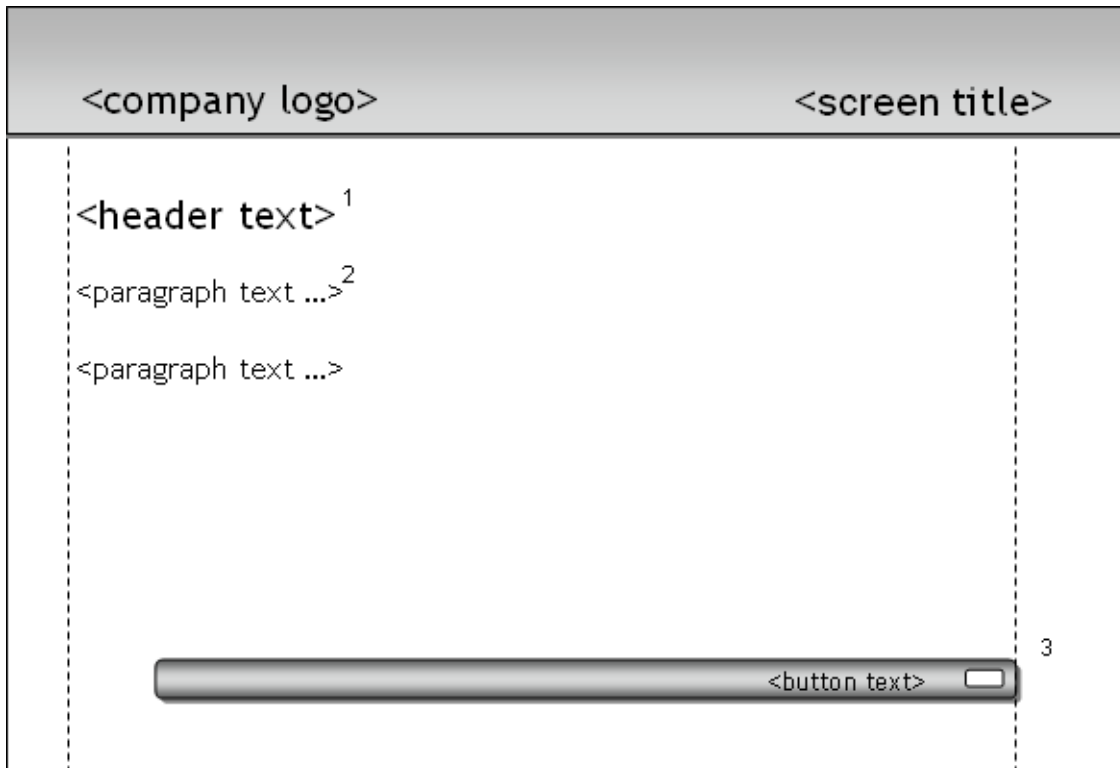
- `Void Clear(Void)`
 - Clear all elements from the search history. Note that this clears the search history for all applications and should be used carefully. The standard usage is to provide a "clear history" button on the search screen, so that it is up to the user when the history stack is cleared.
- `Void Push(String searchTerm)`
 - Push a new search term onto the search history stack.
- `roArray GetAsArray(Void)`
 - Returns the current search history stack as an roArray of Strings with all available search history elements.

Events: None

The roSearchHistory component dispatches no events.

4.9 roParagraphScreen

The Paragraph Screen provides a way to display text and selection choices to the user. This type of screen is frequently used for implementing wizard functionality to guide the user through a specific task. The caller may specify header text which is displayed at the top of the screen and one or more paragraphs of text on the screen. In addition, one or more buttons may be added to the screen to get user input or allow navigation. The screen is designed to automatically format the text, headings and buttons and create the photo-fit for them on screen. Some care must be taken to not provide too much text or clipping may occur.



This object is created with no parameters:

- `CreateObject("roParagraphScreen")`

The `roParagraphScreen` implements the following interfaces:

Interface: `ifParagraphScreen`

`Void AddHeaderText(String text)`

- Add a string of bold, high visibility text to the screen as a header to introduce the subsequent paragraph(s).

`Void AddParagraph(String text)`

- Adds a paragraph of text to the screen. Paragraphs are specified as a single string and they are ordered on the screen in the same order as they are added. Making multiple calls to `AddParagraph()` will continue to add additional paragraphs of text in order until the screen has been filled. The `roParagraphScreen` handles all text formatting and justification. Spacing is automatically inserted between paragraphs for readability.

`Boolean AddButton(Integer id, String title)`

- Adds a button to the screen identified by the title and ID provided. The buttons are displayed in a standard location on the screen and appear in the order added. The ID is defined by the developer and used to uniquely identify the button instance. When the button is pressed, the script will receive an event from the application containing the ID of the button pressed and allowing the script to perform the desired action for that case.

`Void AddGraphic(String url)`

- Adds a graphic image to the screen at the current cursor position and centers it. The current cursor position moves as headers, paragraphs, graphics and buttons are added to the screen. See the Netflix welcome and free trial screens for an example of use.

`Boolean Show(Void)`

- Display or refresh the screen after creation or state changes.

`Void Close(Void)`

- Close the screen and delete the associated object. Useful for avoiding screen flicker when the display order of your screens does not resemble a stack.
- Void SetTitle(String title)
- Set the title for the screen to the specified string.

Since Firmware version 2.6:

- Void AddGraphic(String url, String displayMode)
- Adds a graphic image to the screen at the current cursor position and centers it. The current cursor position moves as headers, paragraphs, graphics and buttons are added to the screen.
 - Sets the mode for displaying the graphic on the screen. This allows images to be either scaled to completely fill the frame (scale-to-fill) or scaled to fit inside the poster frame (scale-to-fit) while maintaining aspect ratio. Valid display modes are:
 - scale-to-fill – scale image to completely fill the rectangle of the bounding frame (Default)
 - scale-to-fit. – scale image to fit horizontally or vertically as appropriate while still maintaining aspect ratio. Note that scale-to-fit may result in pillar-box or letter-box display of images.
 - zoom-to-fill – scales and crops image to maintain aspect ratio and completely fill the rectangle of the bounding frame.
 - photo-fit – Uses several methods to fit the image with a different aspect ratio to the screen. First, it will asymmetrically scale up to a maximum of 5%. Second, for landscape images, if vertical cropping is necessary, it will remove two lines off the bottom for every one line off the top up to a maximum of 30% of the image. For all images, if horizontal cropping is necessary it will crop an equal amount from both sides.

Since Firmware version 2.8:

- Boolean SetDefaultMenuItem(Integer item)
- Set a button id to highlight. Default is the first button. Return true if successful.

Interface: ifSetMessagePort

- Void SetMessagePort(roMessagePort port)
- Set the message port to be used for all events from the screen.

Interface: ifGetMessagePort

- Object GetMessagePort(Void)
- Returns the message port currently set for use by the screen.

Events: roParagraphScreenEvent

The roParagraphScreen sends the roParagraphScreenEvent with the following predicates that indicate its valid event types:

- isButtonPressed()
- Event indicating a button on the screen was selected.
 - Event Details: Type = 5, Msg = "", Index = ID of button from AddButton(), Data = 0
- isScreenClosed()
- Event indicating that the screen was closed and no longer displayed to the user.

- Event Details: Type = 1, Msg = "", Index = 0, Data = 0

Example Code: roParagraphScreen

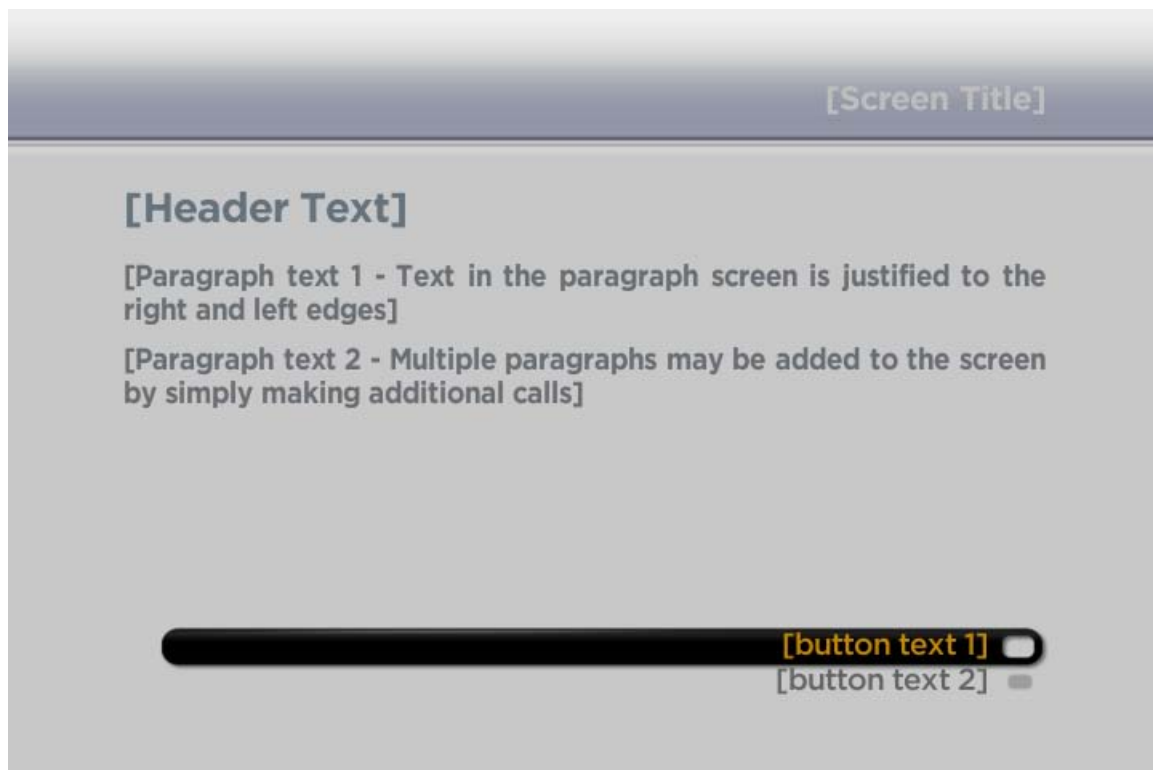
```
Function ShowParagraphScreen() As Void
    port = CreateObject("roMessagePort")
    screen = CreateObject("roParagraphScreen")
    screen.SetMessagePort(port)

    screen.SetTitle("[Screen Title]")
    screen.AddHeaderText("[Header Text]")
    screen.AddParagraph("[Paragraph text 1 - Text in the paragraph
screen is justified to the right and left edges]")
    screen.AddParagraph("[Paragraph text 2 - Multiple paragraphs may be
added to the screen by simply making additional calls]")
    screen.AddButton(1, "[button text 1]")
    screen.AddButton(2, "[button text 2]")
    screen.Show()

    while true
        msg = wait(0, screen.GetMessagePort())

        if type(msg) = "roParagraphScreenEvent"
            exit while
        endif
    end while
End Function
```

Image: roParagraphScreen example results



4.10 roMessageDialog

The Message Dialog displays a formatted, multi-line text message to the user. The dialog may optionally be displayed with a busy animation to indicate progress on a long running operation. The dialog will automatically handle formatting of text and resize to fit. It may also display buttons to get user acknowledgment or a selection choice.

The following example shows an roMessageDialog with a single done button. When the title, text and button are added, the dialog automatically formats and resizes the dialog as needed for display when Show() is called.

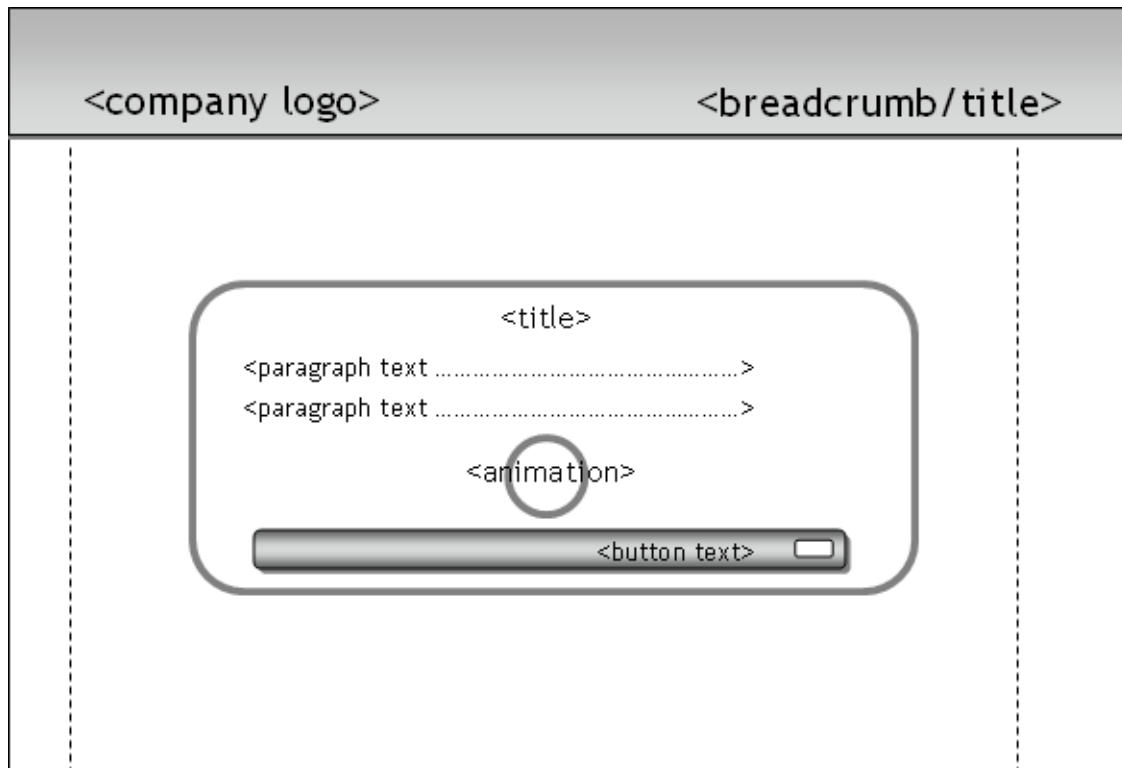


Diagram: roMessageDialog

The roMessage dialog implements the following interfaces:

Interface: ifMessageDialog

- ```
Void SetText(String text)
```
- Set the text to be shown in the message dialog to the specified string. The dialog will automatically resize to accommodate the text provided.
- ```
Boolean AddButton(Integer id, String title)
```
- Adds a button to the screen identified by the title and ID provided. The buttons are at the bottom of the dialog and appear in the order added. When the button is pressed, the script will receive an event from the application indicating the ID of the button pressed.
- ```
Void ShowBusyAnimation(Void)
```
- Display a spinning busy animation to indicate work in progress. The busy animation has currently been defined to be a spinning movie reel. The animation will continue until the screen is closed.
- ```
Boolean Show(Void)
```
- Display or refresh the screen after creation or state changes.

Void Close(Void)

- Close the screen and delete the associated object. Useful for avoiding screen flicker when the display order of your screens does not resemble a stack.

Void SetTitle(String title)

- Set the title for the dialog to the specified string. The title appears in the top center of the dialog in bold text.

Since Firmware version 2.7:

`Void EnableOverlay(Boolean enable)`

- With overlay enabled, the background screen is no longer dimmed.

`Void SetMenuTopLeft(Boolean topLeft)`

- If true, set the format of the buttons to left and top justified. Otherwise default to bottom right justified. Note that if you call `AddRatingButton()`, you must make the buttons top left justified.

`Boolean SetFocusedMenuItem(Integer item)`

- Set a button id to highlight. Default is the first button. Return true if successful.

`Boolean AddRatingButton(Integer id, Integer userRating, Integer aggregateRating, String tip)`

- Adds a star rating button to the dialog. The rating is specified as a string whose value is an integer 1-100 which indicates the number of stars (1 to 5) to be displayed. Think of this as a percentage value $<20\% = 1$ star. This button is displayed in the menu area of the screen and appears in the order added. The `userRating` specifies the value the user rated the title, while the `aggregateRating` represents the total for all users. The `userRating` takes precedence and determines the color of the buttons if set. The `aggregateRating` may display half-stars. The button returns the `userRating` in the event data field.
- The rating widget is only allowed on message dialogs that call `SetMenuTopLeft(true)`.

`Void EnableBackButton(Boolean enableBackButton)`

- Must set `enableBackButton` to true in order to send the `isScreenClosed()` event when the back button is entered.
- By default the `MessageDialog` will not send an `isScreenClosed()` event so that scripts that did not expect this event will not break.

Interface: `ifSetMessagePort`

`Void SetMessagePort(roMessagePort port)`

- Set the message port to be used for all events from the screen.

Interface: `ifGetMessagePort`

`Object GetMessagePort(Void)`

- Returns the message port currently set for use by the screen.

Events: `roMessageDialogEvent`

The `roMessageDialog` sends the `roMessageDialogEvent` with the following predicates that indicate its valid event types:

`isButtonPressed()`

- Event indicating a button on the screen was selected.
- Event Details: Type = 5, Msg = "", Index = ID of button from `AddButton()`, Data = 0

`isScreenClosed()`

- Event indicating that the screen was closed and no longer displayed to the user.
- Event Details: Type = 1, Msg = "", Index = 0, Data = 0

Since Firmware version 2.7:

isButtonInfo()

- Event type indicating the Info button was pressed when the Index=ID button had focus.
- Event Details: Type = 23, Msg = "", Index = ID of button from AddButton(), Data = 0

Example Code: roMessageDialog

The following code example creates a message dialog and displays it to the user. Note that dialogs are not full screen and that the previous screen is dimmed and displays in the background. When the user presses the message dialog button, the dialog is dismissed and the previous screen comes to the foreground.

```
Function ShowMessageDialog() As Void

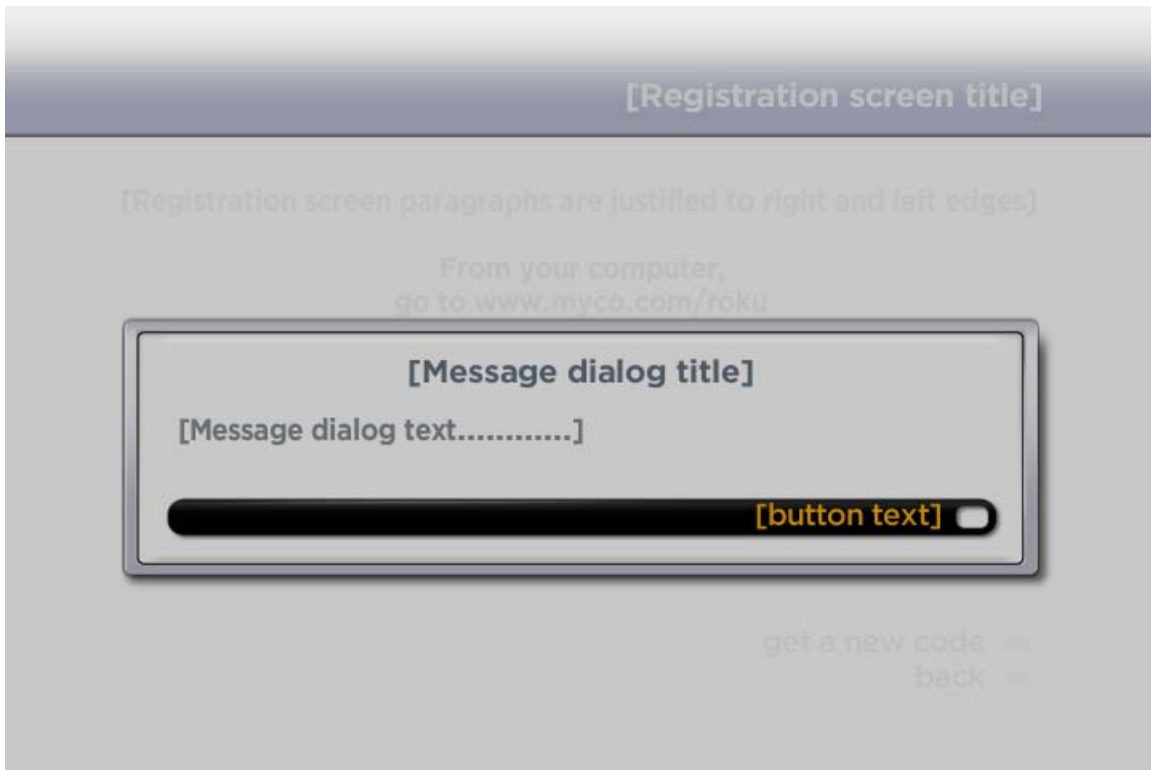
    port = CreateObject("roMessagePort")
    dialog = CreateObject("roMessageDialog")
    dialog.SetMessagePort(port)

    dialog.SetTitle("[Message dialog title]")
    dialog.SetText("[Message dialog text.....]")
    dialog.AddButton(1, "[button text]")
    dialog.Show()

    while true
        dlgMsg = wait(0, dialog.GetMessagePort())
        if type(dlgMsg) = "roMessageDialogEvent"
            if msg.isScreenClosed()
                exit while
            end if
        end if
    end while

End Function
```

Image: roMessageDialog example results



4.11 roOneLineDialog

The One Line Dialog is a special type of dialog optimized for single line text. Unlike the message dialog which displays formatted multi-line messages, the `roOneLineDialog` displays a single line of text centered for the user. This dialog is optimized for rendering of single-line text strings. It is generally used for displaying text to indicate that an operation is in progress. When the operation completes, the dialog is destroyed and the message dialog disappears.

The One Line Dialog implements the following interfaces:

Interface: `ifOneLineDialog`

```
Void SetTitle(String title)
    ▪ Set the title for the screen to the specified string. The title appears in the upper right hand corner of the screen in the overhang area.
Void ShowBusyAnimation(Void)
    ▪ Display a spinning busy animation to indicate work in progress. The busy animation has currently been defined to be a spinning movie reel.
Boolean Show(Void)
    ▪ Display or refresh the screen after creation or state changes.
Void Close(Void)
    ▪ Close the screen and delete the associated object. Useful for avoiding screen flicker when the display order of your screens does not resemble a stack.
```

Interface: `ifSetMessagePort`

```
Void SetMessagePort(roMessagePort port)
    ▪ Set the message port to be used for all events from the screen.
```

Interface: ifGetMessagePort

Object GetMessagePort(Void)

- Returns the message port currently set for use by the screen.

Events: roOneLineDialogEvent

The roOneLineDialog sends the roOneLineDialogEvent with the following predicates that indicate its valid event types:

isButtonPressed()

- Event indicating a button on the screen was selected.
- Event Details: Type = 5, Msg = "", Index = ID of button from AddButton(), Data = 0

isScreenClosed()

- Event indicating that the screen was closed and no longer displayed to the user.
- Event Details: Type = 1, Msg = "", Index = 0, Data = 0

4.12 roCodeRegistrationScreen

The Code Registration Screen is designed to present the user a registration code and the information required to instruct the user on how to register with a service provider. This screen is designed for a rendezvous registration process, where the user is presented a code and the URL for a registration site. The user goes to the site and enters their code, which causes the device and the account to be linked. In the background, the script is polling for completion and the screen is closed to display an activation successful screen when done.

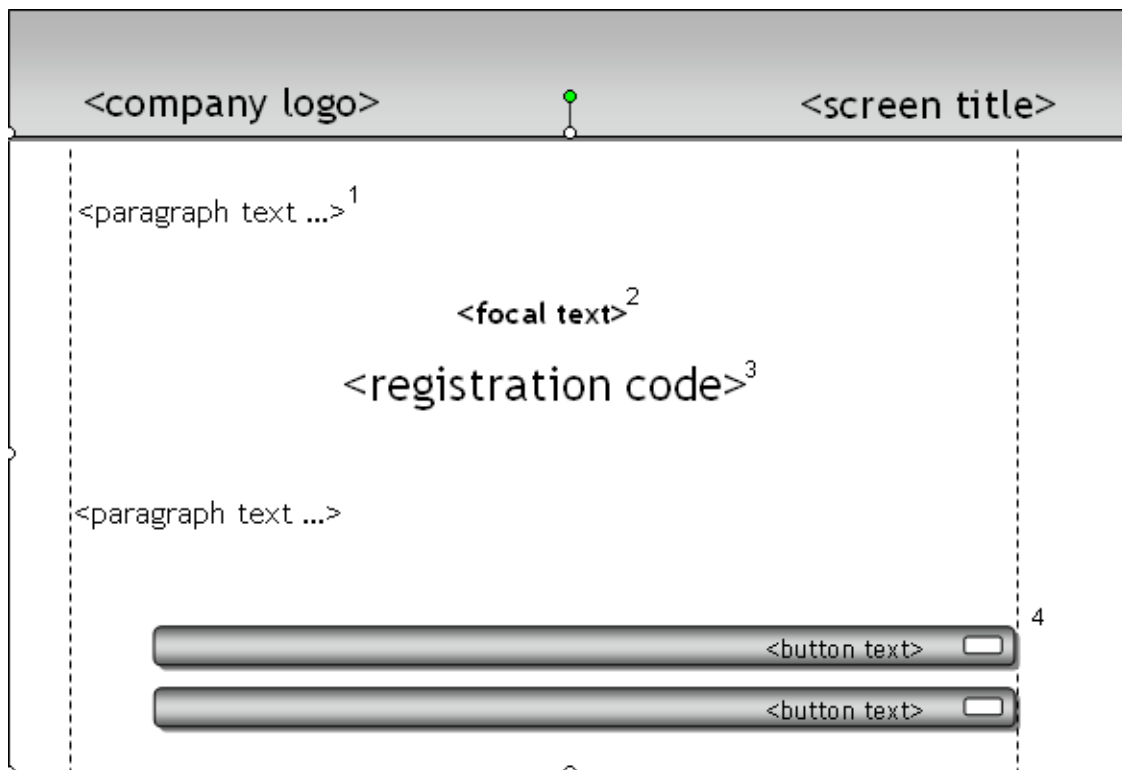


Diagram: roCodeRegistrationScreen

The Code Registration Screen implements the following interfaces:

Interface: `ifCodeRegistrationScreen`

```
Void AddHeaderText(String text)
    ▪ Add a string of bold, high visibility text to the screen as a header to introduce the
      subsequent paragraph(s).
Void AddParagraph(String text)
    ▪ Adds a paragraph of text to the screen. A paragraph is specified as a single string and
      are ordered on the screen in the same order as they are added. The
      roCodeRegistrationScreen handles all text formatting and justification. Spacing is
      automatically inserted between paragraphs for readability.
Boolean AddButton(Integer id, String title)
    ▪ Adds a button to the screen identified by the title and ID provided. The buttons are at the
      bottom of the screen and appear in the order added. When the button is pressed, the
      script will receive an event from the application indicating the ID of the button pressed.
Void AddFocalText(String text, String spacingFormat)
    ▪ Adds high visibility focal text to the screen to be placed above the registration code. This
      text is intended to provide the user important instructions on where to use the registration
      code. It is generally a few words of instruction followed by the URL for the registration
      site on the web. Multiple lines of text may be added and the spacing between each is
      controlled by specifying the spacing format as one of the following: spacing-dense,
      spacing-normal or spacing-sparse.
Void SetRegistrationCode(String regCode)
    ▪ Set the registration code (e.g. XM3RT) or text (e.g. retrieving...) to be displayed on the
      screen.
Boolean Show(Void)
    ▪ Display or refresh the screen after creation or state changes.
Void Close(Void)
    ▪ Close the screen and delete the associated object. Useful for avoiding screen flicker
      when the display order of your screens does not resemble a stack.
Void SetTitle(String title)
    ▪ Set the title for the screen to the specified string.
```

Interface: `ifSetMessagePort`

```
Void SetMessagePort(roMessagePort port)
    ▪ Set the message port to be used for all events from the screen.
```

Interface: `ifGetMessagePort`

```
Object GetMessagePort(Void)
    ▪ Returns the message port currently set for use by the screen.
```

Events: `roCodeRegistrationScreenEvent`

The `roCodeRegistrationScreen` sends the `roCodeRegistrationScreenEvent` with the following predicates that indicate its valid event types:

```
isButtonPressed()
    ○ Event indicating a button on the screen was selected.
    ○ Event Details: Type = 5, Msg = "", Index = ID of button from AddButton(), Data = 0
isScreenClosed()
    ○ Event indicating that the screen was closed and no longer displayed to the user.
    ○ Event Details: Type = 1, Msg = "", Index = 0, Data = 0
```

Example Code: roCodeRegistrationScreen

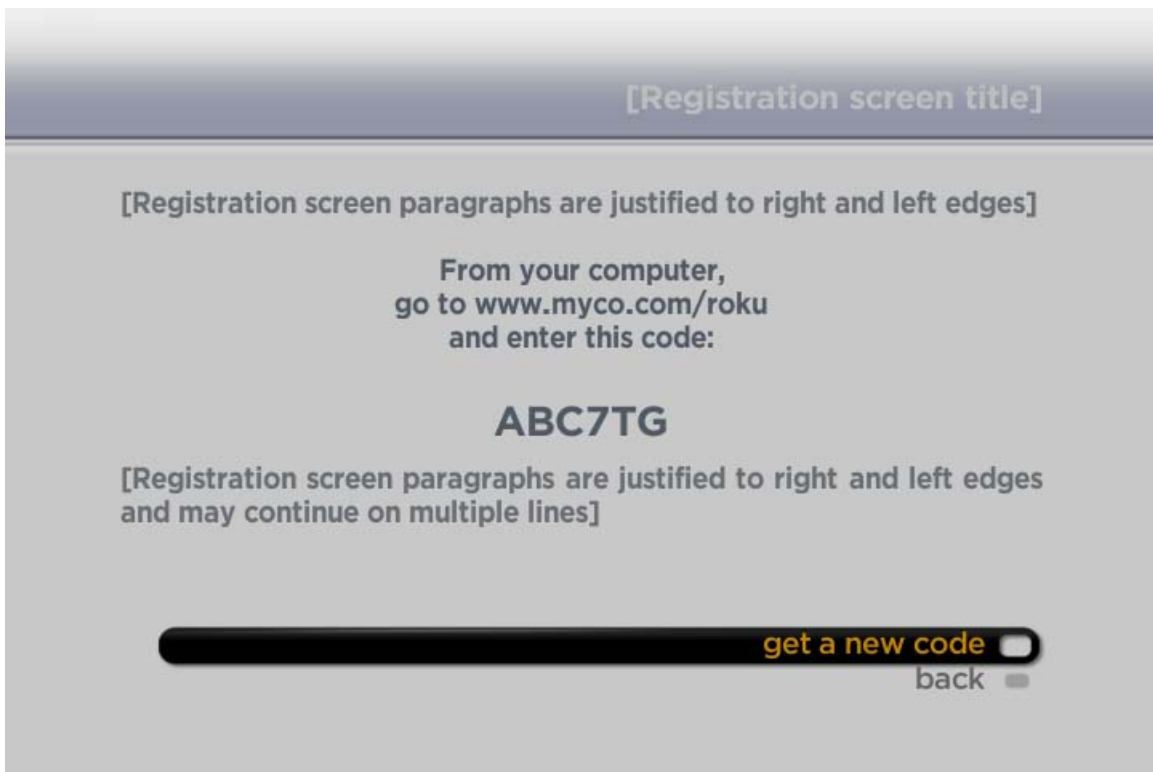
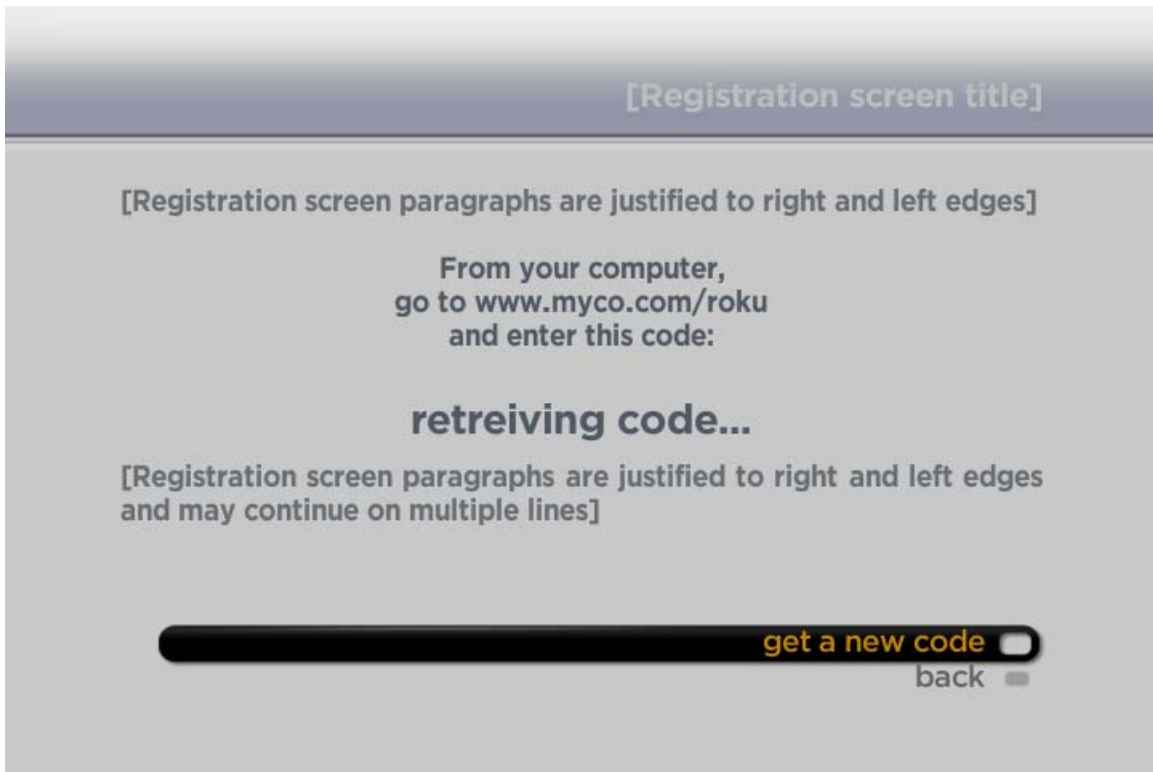
```
Function ShowMessageDialog() As Void
    port = CreateObject("roMessagePort")
    screen = CreateObject("roCodeRegistrationScreen")
    screen.SetMessagePort(port)

    screen.SetTitle("[Registration screen title]")
    screen.AddParagraph("[Registration screen paragraphs are justified
to right and left edges]")
    screen.AddFocalText(" ", "spacing-dense")
    screen.AddFocalText("From your computer,", "spacing-dense")
    screen.AddFocalText("go to www.myco.com/roku", "spacing-dense")
    screen.AddFocalText("and enter this code:", "spacing-dense")
    screen.AddFocalText(" ", "spacing-dense")
    screen.SetRegistrationCode("retrieving code...")
    screen.AddParagraph("[Registration screen paragraphs are justified
to right and left edges and may continue on multiple lines]")
    screen.AddButton(0, "get a new code")
    screen.AddButton(1, "back")
    screen.Show()

    sleep (10000) 'simulate fetching registration code from webapi
    screen.SetRegistrationCode("ABC7TG")
    screen.Show()

    while true
        dlgMsg = wait(0, dialog.GetMessagePort())
        exit while
    end while
End Function
```

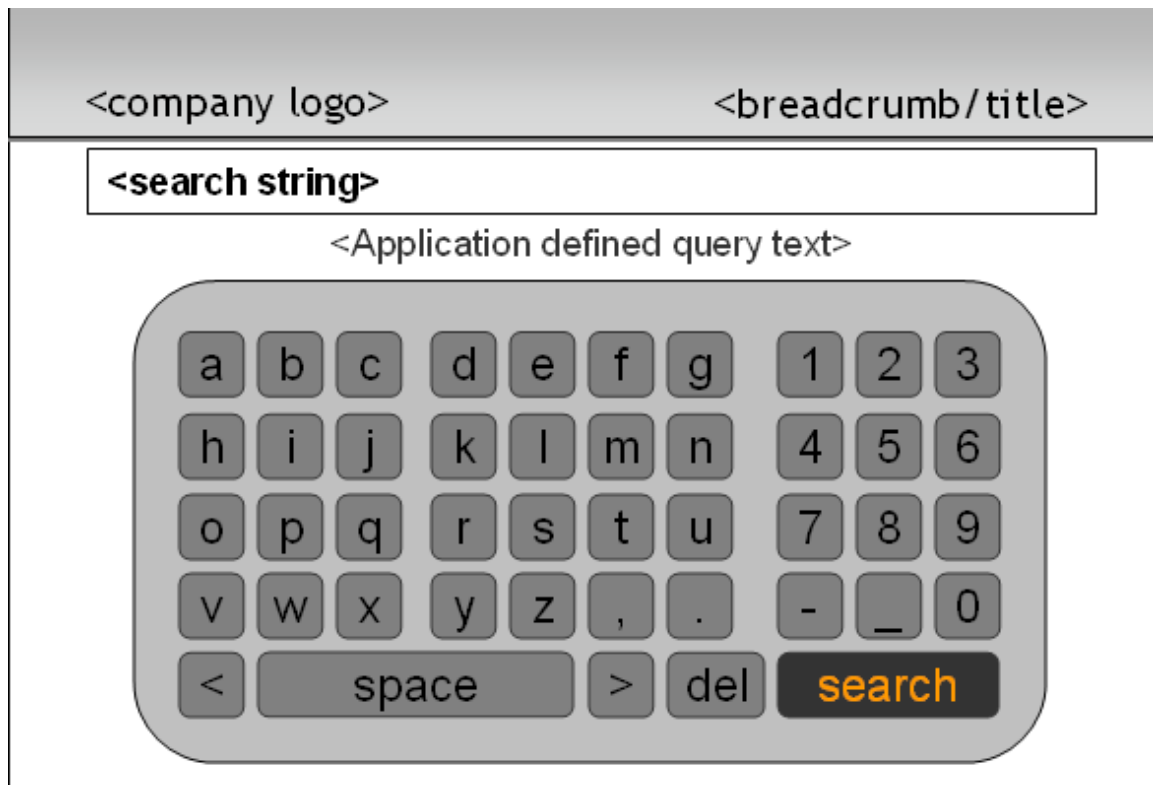
Image: roCodeRegistrationScreen example results



4.13 roKeyboardScreen

The Keyboard Screen is designed to allow the user to enter an alpha-numeric string for searching, username/password registration or other purposes. It is generally used as part of a sequence of screens and the results are displayed on the subsequent screen in the sequence. In the case of a search screen, results are displayed on the roPosterScreen and categories may be used to segregate TV and Movie results.

Diagram: roKeyboardScreen



Interface: ifKeyboardScreen

The roKeyboardScreen implements the following interfaces:

```
Void SetDisplayText(String displayText)
    ▪ Sets the descriptive text displayed to the user for prompting regarding the required entry.
Void SetText(String title)
    ▪ Sets the default text displayed in the keyboard entry field. Text is selected by default, so
      that if the user begins to enter text, the default text is replaced. If the user presses the
      action button (e.g. Search), the default text is used as the entry.
String GetText(Void)
    ▪ Returns the current value in the keyboard text entry field.
Void SetMaxLength(Integer maxLen)
    ▪ Set the maximum length for text entry. Default: 20 characters.
Void AddButton(Integer id, String title)
```

- Adds a button to the screen identified by the title and ID provided. The buttons are at the bottom of the screen and appear in the order added. When the button is pressed, the script will receive an event from the application indicating the ID if the button pressed.
- ```
Void setSecureText(Boolean isSecure)
```
- If isSecure is true the user entered text is obscured with bullet characters. If false, it's treated as plain text. The default is false.
- ```
Boolean Show(Void)
```
- Display or refresh the screen after creation or state changes.
- ```
Void Close(Void)
```
- Close the screen and delete the associated object. Useful for avoiding screen flicker when the display order of your screens does not resemble a stack.
- ```
Void SetTitle(String title)
```
- Set the title for the screen to the specified string.

Interface: ifSetMessagePort

- ```
Void SetMessagePort(roMessagePort port)
```
- Set the message port to be used for all events from the screen.

#### Interface: ifGetMessagePort

- ```
Object GetMessagePort(Void)
```
- Returns the message port currently set for use by the screen.

Events: roKeyboardScreenEvent

The roKeyboardScreen sends the roKeyboardScreenEvent with the following predicates that indicate its valid event types:

- ```
isButtonPressed()
```
- Event indicating a button on the screen was selected and the ID of the button that was selected.
  - Event Details: Type = 5, Msg = "", Index = ID set in AddButton() request, Data = 0.
- ```
isScreenClosed()
```
- Event indicating that the screen was closed and no longer displayed to the user.
 - Event Details: Type = 1, Msg = "", Index = 0, Data = 0.

Example Code: roKeyboardScreen

```
Sub Main()

    screen = CreateObject("roKeyboardScreen")
    port    = CreateObject("roMessagePort")

    screen.SetMessagePort(port)
    screen.SetTitle("Search Screen")
    screen.SetText("default")
    screen.SetDisplayText("enter text to search")
    screen.SetMaxLength(8)
    screen.AddButton(1, "finished")
```

```

screen.AddButton(2, "back")
screen.Show()

while true
    msg = wait(0, screen.GetMessagePort())

    print "message received"
    if type(msg) = "roKeyboardScreenEvent"
        if msg.isScreenClosed()
            return

        else if msg.isButtonPressed() then
            print "Evt: "; msg.GetMessage      (); " idx: "; msg.GetIndex()
            if msg.GetIndex() = 1
                searchText = screen.GetText()
                print "search text: "; searchText
                return
            endif
        endif
    endif
end while

End Sub

```

Image: roKeyboardScreen example results



4.14 roPinEntryDialog

The Pin Entry Dialog is designed to allow the user to enter a numeric PIN for purchasing content. Users establish a PIN on the partner's website for purchasing transactions. The roPinEntryDialog allows the script to present the user with a pop-up, modal dialog for PIN entry and then the script can subsequently call the API's to conclude the purchase transaction. When the last digit is entered, focus jumps to the first button.

Image: roPinEntryDialog sample



Diagram: roPinEntryDialog

The roPinEntryDialog implements the following interfaces:

Interface: ifPinEntryDialog

`Boolean AddButton(Integer id, String title)`

- Adds a button to the screen identified by the title and ID provided. The buttons are at the bottom of the screen and appear in the order added. When the button is pressed, the script will receive an event from the application indicating the ID of the button pressed.

`String Pin(Void)`

- Returns a string containing the PIN entered by the user.

`Void SetNumPinEntryFields(Integer numFields)`

- Sets the maximum number of digits to be entered and displayed for the pin.

`Boolean Show(Void)`

- Display or refresh the screen after creation or state changes.

`Void Close(Void)`

- Close the screen and delete the associated object. Useful for avoiding screen flicker when the display order of your screens does not resemble a stack.

`Void SetTitle(String title)`

- Set the title for the screen to the specified string.

Since Firmware version 3.0:

Void EnableBackButton(Boolean enableBackButton)

- Must set enableBackButton to true in order to send the isScreenClosed() event when the back button is entered.
- By default the PinEntryDialog will not send an isScreenClosed() event so that scripts that did not expect this event will not break.

Interface: ifSetMessagePort

Void SetMessagePort(roMessagePort port)

- Set the message port to be used for all events from the screen.

Interface: ifGetMessagePort

Object GetMessagePort(Void)

- Returns the message port currently set for use by the screen.

Events: roPinEntryDialogEvent

The roPinEntryDialog sends the roPinEntryDialogEvent with the following predicates that indicate its valid event types:

isButtonPressed()

- Event indicating a button on the screen was selected.
- Event Details: Type = 5, Msg = "", Index = ID of button from AddButton(), Data = 0

isScreenClosed()

- Event indicating that the screen was closed and no longer displayed to the user.
- Event Details: Type = 1, Msg = "", Index = 0, Data = 0

4.15 roAudioPlayer

The Audio Player object provides the ability to setup the playing of a series of audio streams. Audio streams may be WMA or MP3 encoded. The object accepts an array of content meta-data objects, describing the audio and providing url's for accessing each stream. This object does not provide an interface to a screen. In order to get events from the UI you are using and the Audio Player you should use a common Message Port for both objects.

This object is created with no parameters:

- CreateObject("roAudioPlayer")

The Audio Player implements the following interfaces:

Interface: ifAudioPlayer

Void SetContentList(Array contentList)

- Set the content to be played by the Audio Player. The caller passes an Array of AssociativeArrays (Content Meta-Data objects) representing the information for each stream to be played. See "Content Meta-Data" for details on the attributes for each element in the array.

Void AddContent(AssociativeArray contentItem)

- Add a new ContentMetaData item to the content list for the Audio Player. New items are added to the end of the list.

```
Void ClearContent(Void)
```

- Clear all content from the Audio Player.

```
Bool Play(Void)
```

- Put the Audio Player into *play* mode starting at the beginning of the Content List.
- This will stop any currently playing Content List.

```
Bool Stop(Void)
```

- Stop Audio Player from playing or pausing and cleanup.

```
Bool Pause(Void)
```

- Put Audio Player into *pause* mode.
- It is an error to Pause if player is not in play mode.

```
Bool Resume(Void)
```

- Put Audio Player into *play* mode starting from the pause point.
- It is an error to Resume from any other mode than Pause.

```
Void SetLoop(Boolean enable)
```

- Enable/Disable the automatic replaying of the Content List.

```
Void SetNext(Integer item)
```

- Set what the next item to be played within the Content List should be.

Since Firmware version 2.6:

```
Boolean Seek(Integer OffsetMs)
```

- Set the start point of playback for the current song to OffsetMs milliseconds.

Interface: ifHttpAgent

The ifHttpAgent methods setup the way URLs in the Content Meta-Data are accessed via this object.

```
Boolean AddHeader(String name, String value)
```

- Add the specified HTTP header.
- If "x-roku-reserved-dev-id" is passed as name, the Roku box ignores the passed in value. In its place, sends the devid of the application publisher of the currently running app. This allows the developer's server to know which client app is talking to it.
- Developers may set any headers except what's explicitly reserved by Roku. The Roku box will enforce ownership of values in the namespace x-roku-reserved-
- The Roku box will set initial values of well known headers (e.g. User-Agent, Content-Length, etc.)
- Users may override well known values if needed (e.g. some servers may require a specific user agent string)
- The Roku box will validate the data set in the headers to ensure it's of the right type, not too long, etc.

```
Boolean SetHeaders(roAssociativeArrayString nameValueMap)
```

- Add the specified HTTP headers in the nameValueMap associative array.
- Header limitations specified in AddHeader() still apply.

```
Boolean SetCertificatesFile(String path)
```

- Set the certificates file used for SSL to the .pem file specified. The .pem file should include the ca (certificate authority) cert that signed the cert installed on your web server. This enables authentication of your server.
- Instances of roUrlTransfer and ifHttpAgent components should call this function before performing https requests.
- The appropriate certificates file should be placed at the location specified in the SetCertificatesFile() function call.

```
Boolean InitClientCertificates(Void)
```

- Initialize the object to send the Roku client cert.

Example Code: ifHttpAgent

```
ba = CreateObject("roByteArray")
ba.FromAsciiString("myusername:mypassword")
myhttpagent.AddHeader("Authorization", "Basic " + ba.ToBase64String())
```

Interface: ifSetMessagePort

Void SetMessagePort(roMessagePort port)

- Set the message port to be used for all events from the Audio Player.

Interface: ifGetMessagePort

Object GetMessagePort(Void)

- Returns the Message Port currently set for use by the Audio Player.

Events: roAudioPlayerEvent

The roAudioPlayer sends the roAudioPlayerEvent with the following predicates that indicate its valid event types:

isRequestSucceeded()

- Event to indicate successful completion of stream playback.
- Event Details: Type = 8, Msg = "", Index = <Index of audio stream>, Data = 0

isRequestFailed()

- Event to indicate error in audio playback
- Event Details: Type = 9, Msg = <text description of error>, Index = <error id>, Data = 0
 - Error number codes:
 - 0 = Network error : server down or unresponsive, server is unreachable, network setup problem on the client.
 - -1 = HTTP error: malformed headers or HTTP error result.
 - -2 = Connection timed out
 - -3 = Unknown error

isListItemSelected()

- Event indicating that a stream has been selected to start playing.
- Event Details: Type=0, Msg = "", Index=Index of currently selected item, Data = 0

isStatusMessage()

- Event with status messages about playback.
- Event Details: Type = 11, Msg = <audio player defined messages>, Index = "", Index=0, Data = 0
- <audio player defined messages> from Msg above:
 - **end of stream**– Event sent when a stream is finished playing (deprecated)
 - **end of playlist**– Event sent when Content List has completed and stopped (deprecated)
 - **start of play** – Event sent when stream has started playing
 - **startup progress** – Event sent during the startup phase of a stream

isFullResult()

- Event to indicate that playback completed at end of content
 - Event Details: Event Details: Type = 16, Msg = "", Index = 0, Data = 0
- isPartialResult()**
- Event to indicate playback was interrupted
 - Event Details: Type = 15, Msg = "", Index = 0, Data = 0

Since Firmware version 2.6:

- isPaused()**
- Event indicating that video playback was paused by the user.
 - Event Details: Type = 12, Msg = "", Index = 0, Data = 0
- isResumed()**
- Event indicating that video playback has resumed.
 - Event Details: Type = 13, Msg = "", Index = 0, Data = 0

Example Code: roAudioPlayer

```
Sub Main()  
    audioPlayer = CreateObject("roAudioPlayer")  
    port = CreateObject("roMessagePort")  
    audioPlayer.SetMessagePort(port)  
    song = CreateObject("roAssociativeArray")  
    song.url = "http://www.the flute.co.uk/media/BachCPE_SonataAmin_1.wma"  
    audioplayer.addcontent(song)  
    audioplayer.setloop(false)  
    audioPlayer.play()  
    while true  
        msg = wait(0, port)  
        if type(msg) = "roAudioPlayerEvent"  
            if msg.isStatusMessage() then  
                print "roAudioPlayerEvent: "; msg.getMessage()  
                if msg.getMessage() = "end of playlist" return  
            endif  
        endif  
    end while  
End Sub
```

4.16 roMessagePort

A Message Port is the place messages (events) are sent. See the “Event Loops” section for more details. When using BrightScript, you would not call these functions directly. Instead, use the “Wait” BrightScript statement (see the BrightScript documentation).

This object is created with no parameters:

- `CreateObject("roMessagePort")`

Interface: ifMessagePort

```
roMessagePort GetMessage (Void)  
roMessagePort WaitMessage(Integer timeout)
```

4.17 roDeviceInfo

The Device Info provides an interface to obtain attributes about the device. These attributes are not changeable by the script, but may be queried to obtain values for the player. It is common for scripts to need access to the serial number and model info for negotiating with backend services.

This object is created with no parameters:

- `CreateObject("roDeviceInfo")`

The roDeviceInfo implements the following interfaces:

Interface: ifDeviceInfo

```
String GetModel(Void)  
    ▪ Returns the model name for the Roku Streaming Player device running the script as  
      a string. For example "N1000". Please see section 1.4 of the Developer Guide for a  
      list of the current and classic models.  
String GetVersion(Void)
```

- Returns the version number of the Roku Streaming Player firmware running on the device in display format. For example "1.5 build 909".

String GetDeviceUniqueId(Void)

- Returns an identifier that is unique to the unit running the script.
- On the Roku Streaming Player, this string is the device serial number which is burned into the unit at manufacturing time in the form of PRRXXXXXXXXX. Where P=Prefix, RR=H/W Revision, X=Device Unique Digit Sequence For example: 20E825000036.

String GetDisplayType(Void)

- Returns the text corresponding to the button selection in the Player Info Settings/Display Type page. Either "HDTV", "4x3 standard", or "16x9 anamorphic"

String GetDisplayMode(Void)

- Returns "720p" or "480i"

String GetDisplayAspectRatio(Void)

- Returns "4x3" or "16x9"

Since Firmware version 2.6:

roAssociativeArray GetDisplaySize(Void)

- Returns an roAssociativeArray with keys "w" and "h" that contain the values for the screen width and height respectively. Example: { w:1080, h:720 }

String GetTimeZone(Void)

- Returns a string representing the current system time zone selection.

Since Firmware version 2.7:

Boolean HasFeature(String feature)

- Returns true if the current device/firmware supports the passed in feature string
- Valid features to query for are:
 - "5.1_surround_sound"
 - "sd_only_hardware"
 - "usb_hardware"
 - "1080p_hardware"

Since Firmware version 2.8:

roAssociativeArray GetIPAddr()

- Returns roAssociative array of device interface / ipaddress name/value string pairs
- Provides a way for your application to get the local ip address of the roku box that can be used in conjunction with the ECP (see the External Control Protocol Guide) command "**launch**" to start a different channel from the current channel.

Example: roDeviceInfo

```
di = CreateObject("roDeviceInfo")
print di.GetModel()
print di.GetVersion()
print di.GetDeviceUniqueId()
```

On a particular system this generates:

```
N1000
999.99E99999X
20E825000036
```

4.18 roDateTime

The Date Time provides an interface to obtain the current date/time for the player. The date time may be initialized either from the system time, seconds from epoch or a valid date/time string. This component provides several options for obtaining attributes about the date/time. All times are GMT unless they are converted to the system timezone with a call to the method: `toLocalTime()`.

This object is created with no parameters:

- `CreateObject("roDateTime")`

The `DateTime` implements the following interfaces:

Interface: `ifDateTime`

```
Integer asSeconds(Void)
    • Returns the current date/time as the number of seconds from epoch (1/1/1970)
String asDateString(String format)
    • Returns the current date/time as a formatted string in one of the following formats:
        ▪ "long-date" - e.g. Tuesday October 9, 2001
        ▪ "short-date" – e.g. 1/1/2009
        ▪ "short-month" – e.g. Jan
        ▪ "short-weekday" – e.g. Mon
String asDateStringNoParam(Void)
    • Returns the current date/time formatted as the default "long-date"
String getWeekday(Void)
    • Returns the current day of the week as a String (e.g. Monday).
Integer getDayOfMonth(Void)
    • Returns the current day of the month as an Integer (e.g. 15 for the 15th)
Integer getLastDayOfMonth(Void)
    • Returns the last day of the current month as an Integer.
Integer getMonth(Void)
    • Returns the current month as an Integer (1=Jan, 12=Dec).
Integer getYear(Void)
    • Returns the current year as an Integer..
Void mark(Void)
    • Initialize the date/time value to the current time.
Void fromSeconds(Integer numSeconds)
    • Initialize the date/time value using the number of seconds from epoch.
Void fromISO8601String(String dateString)
    • Initialize the date/time using a string in the ISO8601 format. For example "YYYY-MM-DD HH:MM:SS" e.g "2009-01-01 01:00:00.000" or "2009-01-01T01:00:00.000". Note that this function is unaware of the local time zone, so these time formats are effectively UTC even though the 8601 spec says they should be in local time. The above formats are also the only formats recognized by this function, even though the 8601 spec contains other valid formats.
```

Since Firmware version 2.6:

```
Integer getHours(Void)
    • Returns the current hour as an Integer.
Integer getMinutes(Void)
    • Returns the current minute as an Integer.
```

```
Integer getSeconds(Void)
    • Returns the current second as an Integer.
Integer getMilliseconds(Void)
    • Returns the current millisecond as an Integer
Void toLocalTime(Void)
    • Converts the current time to the current local time using the system time zone setting.
      This function is not idempotent, and multiple calls will do multiple timezone
      adjustments to the time yielding an incorrect result.
```

4.19 roEVP cipher

The EVP Cipher component provides an interface to the OpenSSL EVP library of symmetric cipher commands. The EVP library provides a high-level interface to cryptographic functions to implement digital “envelopes”. These commands allow data to be encrypted or decrypted using various block and stream ciphers using keys based on passwords or explicitly provided. For additional information on roByteArray used with the roEVP cipher see the BrightScript Reference Manual.

Some of the ciphers do not have large keys and others have security implications if not used correctly. A beginner is advised to just use a strong block cipher in CBC mode such as bf or des3. All the block ciphers normally use PKCS#5 padding also known as standard block padding: this allows a rudimentary integrity or password check to be performed. However since the chance of random data passing the test is better than 1 in 256 it isn't a very good test. If padding is disabled then the input data must be a multiple of the cipher block length. All RC2 ciphers have the same key and effective key length. Blowfish and RC5 algorithms use a 128 bit key.

For additional information on the OpenSSL library of symmetric ciphers see:

<http://www.openssl.org/docs/apps/enc.html>

The roEVP cipher implements the following interfaces:

Interface: ifEVP cipher

```
Integer Setup(Boolean, String, String, String, Integer)
    • Setup and initialize a new cipher context. The Setup function takes the following
      parameters:
        • Boolean - true for encrypt, false for decrypt
        • String - cipher format string, from openssl, listed below
        • String - hex-encoded key
        • String - hex-encoded initialization vector (can be empty string)
        • Integer - 1 to use standard padding, 0 for no padding
Integer Reinit(Void)
    • Reinitialize an existing cipher context.
roByteArray Process(roByteArray)
    • Process data in the roEVP cipher context. Returns roByteArray containing the
      results.
roByteArray Update(roByteArray)
    • Update context with additional bytes from roByteArray provided.
roByteArray Final(Void)
    • Returns roByteArray with final results of cipher operation.
```

List of Supported Ciphers:

| | |
|------------------------|---|
| blowfish | Blowfish |
| bf-cbc | Blowfish in CBC mode |
| bf | Alias for bf-cbc |
| bf-cfb | Blowfish in CFB mode |
| bf-ecb | Blowfish in ECB mode |
| bf-ofb | Blowfish in OFB mode |
| des-cbc | DES in CBC mode |
| des | Alias for des-cbc |
| des-cfb | DES in CFB mode |
| des-ofb | DES in OFB mode |
| des-ecb | DES in ECB mode |
| des-ede-cbc | Two key triple DES EDE in CBC mode |
| des-ede | Two key triple DES EDE in ECB mode |
| des-ede-cfb | Two key triple DES EDE in CFB mode |
| des-ede-ofb | Two key triple DES EDE in OFB mode |
| des-ede3-cbc | Three key triple DES EDE in CBC mode |
| des-ede3 | Three key triple DES EDE in ECB mode |
| des3 | Alias for des-ede3-cbc |
| des-ede3-cfb | Three key triple DES EDE CFB mode |
| des-ede3-ofb | Three key triple DES EDE in OFB mode |
| desx | DESX algorithm. |
| desx-cbc | desx in cbc mode |
| aes-[128 192 256]-cbc | 128/192/256 bit AES in CBC mode |
| aes-[128 192 256] | Alias for aes-[128 192 256]-cbc |
| aes-[128 192 256]-cfb | 128/192/256 bit AES in 128 bit CFB mode |
| aes-[128 192 256]-cfb1 | 128/192/256 bit AES in 1 bit CFB mode |
| aes-[128 192 256]-cfb8 | 128/192/256 bit AES in 8 bit CFB mode |
| aes-[128 192 256]-ecb | 128/192/256 bit AES in ECB mode |
| aes-[128 192 256]-ofb | 128/192/256 bit AES in OFB mode |

4.20 roEVPDigest

The EVP Digest component provides an interface to the OpenSSL EVP library of message digest algorithms. The EVP library provides a high-level interface to cryptographic functions to implement digital “envelopes”. The digest of choice for all new applications is SHA1. Other digests are however still widely used.

For additional information on the OpenSSL library of message digest algorithms see:

<http://www.openssl.org/docs/apps/dgst.html>

The roEVPDigest implements the following interfaces:

Interface: ifEVPDigest

Integer Setup(String digestType)

- Initialize a new message digest context for processing, setting the digest type to the string for one of the supported digest algorithms.

Integer Reinit(Void)

- Re-initialize an existing message digest context.
- ```
String Process(String)
```
- Process data in the current context.
- ```
Void Update(String)
```
- Update/add data to the current context for processing.
- ```
String Final(Void)
```
- Return final results from the message digest operation.

#### List of Supported Digest Algorithms:

```
md5 md5 message digest algorithm (default)
md2 md2 message digest algorithm
sha1 sha1 message digest algorithm
sha sha message digest algorithm
sha224, sha256, sha384, sha512
```

#### Example: SHA1 Message Digest with roEVPDigest

```
ba = CreateObject("roByteArray")
...populate bytearray...
digest = CreateObject("roEVPDigest")
digest.Setup("sha1")
result = digest.Process(ba)
print result
```

#### Example: MD5 Message Digest with roEVPDigest

```
ba1 = CreateObject("roByteArray")
...populate ba1...
ba2 = CreateObject("roByteArray")
ba2.FromAsciiString(somestring)
digest = CreateObject("roEVPDigest")
digest.Setup("md5")
digest.Update(ba1)
digest.Update(ba2)
result = digest.Final()
print result
```

## 4.21 roHMAC

The HMAC component provides an interface to the OpenSSL HMAC functions. For additional information on the OpenSSL HMAC functions, please see:

<http://www.openssl.org/docs/crypto/hmac.html>

#### Interface: ifHMAC

```
Integer Setup(String digestType, roByteArray key)
```

- Initialize new HMAC context for processing, setting the digest type to the string for one of the supported digest algorithms and the key to the passed roByteArray. Returns 0 on success, -1 on failure.

Integer Reinit(Void)

- Re-initialize an existing HMAC context.

roByteArray Process(roByteArray message)

- Process message data in the HMAC context. Returns an roByteArray containing the results.

Void Update(roByteArray partialMessage)

- Update/add message data to the HMAC context for processing. No return value

roByteArray Final(Void)

- Return an roByteArray with the final results of the HMAC operation.

### List of Supported Digest Algorithms:

|                                |                                        |
|--------------------------------|----------------------------------------|
| md5                            | md5 message digest algorithm (default) |
| md2                            | md2 message digest algorithm           |
| sha1                           | sha1 message digest algorithm          |
| sha                            | sha message digest algorithm           |
| sha224, sha256, sha384, sha512 |                                        |

### Example: HMAC usage

```
hmac = CreateObject("roHMAC")
signature_key = CreateObject("roByteArray")
signature_key.fromAsciiString(getKey())
if hmac.setup("sha1", signature_key) = 0
 message = CreateObject("roByteArray")
 message.fromAsciiString(getMessage())
 result = hmac.process(message)
 print result.toBase64String()
end if

hmac = CreateObject("roHMAC")
signature_key = CreateObject("roByteArray")
signature_key.fromAsciiString(getKey())
if hmac.setup("sha1", signature_key) = 0
 message1 = CreateObject("roByteArray")
 message1.fromAsciiString(getMessage1())
 hmac.update(message1)

 message2 = CreateObject("roByteArray")
 message2.fromAsciiString(getMessage2())
 hmac.update(message2)

 result = hmac.final()
 print result.toBase64String()
end if
```

## 4.22 roTimespan

The Timespan object provides an interface to a simple timer for tracking the duration of activities. It's useful for tracking how an action has taken or if a specified time has elapsed from some starting event.



The roTimespan implements the following interfaces:

#### Interface: ifTimespan

```
Void Mark(Void)
 ▪ Sets the initial "Mark" point for tracking a timespan.
Integer TotalMilliseconds(Void)
 ▪ Returns the total number of Milliseconds from the "Mark" point
Integer TotalSeconds(Void)
 ▪ Returns the total number of seconds from the "Mark" point.
Integer GetSecondsToISO8601Date(String date)
 ▪ This function parses the ISO8601 date (e.g. 2008-11-29T14:54:02.171) and returns
 the number of seconds from now until the specified date/time. The date provided and
 the current time calculations are all done assuming UTC. The "Z" timezone part of an
 ISO8601 string does not currently work correctly.
```

#### Example: roTimespan (timing an activity)

```
REM *****
REM Compute the number of millisecs to perform a task
REM *****
timer = CreateObject("roTimespan")
timer.Mark()
DoTimeConsumingTask()
Print "Task took: " + itostr(timer.TotalMilliseconds())

REM *****
REM Compute how many seconds until rental expires
REM *****
Function secondsLeft(String expirationDate) As Integer
 str = expirationDate
 if str = invalid return -1
 ts = CreateObject("roTimespan")
 seconds = ts.GetSecondsToISO8601Date(str)
 print "Expires: " + str + " secs: " + itostr(seconds)
 return seconds
End Function
```

## 4.23 roRegistry

The Registry is an area of non-volatile storage where a small number of persistent settings can be stored. A single developer's registry can be shared across multiple applications by using the same set of keys to encrypt them. This is the conventional way that an "application suite" with shared preferences and other shared information should work. The registry is encrypted, and updates are relatively performance intensive and should be used sparingly. Note that all writes to the registry are delayed, and not committed to non-volatile storage until an explicit Flush() is done. The platform may choose opportune times to flush data on its own, but no application is technically correct unless it explicitly calls Flush() at appropriate times in their application.

The maximum size of registry data is 16K per suite. Care should be taken to minimize the amount of data stored and the frequency it is updated. When the registry is flushed, the registry for the application is re-written, potentially resulting in a performance hit. The Registry data is stored in a fault tolerant manner by preserving a backup for each write which is automatically rolled-back in the event of a failure.

The system also maintains a registry for common system data. This registry is not accessible by the application except through BrightScript Component API's. Any elements that may be modifiable by the script are exposed through API's to the application.

The Registry is divided into sections which are specified by the developer for organization and grouping of attributes. API's are provided to list the sections in the registry and to provide access to the data in each section.

The Registry also supports the use of a special "Transient" registry section. This registry section allows the application to store attributes that have the lifetime of a single boot. Within a specific boot session, these values will be persistent to the application and stored as any other registry value. Whenever the user reboots the Roku Streaming Player, the "Transient" registry section is removed and the values no longer persist. This technique is useful for caching data to minimize network access, yet still ensuring that this data is always fresh after a system reboot.

Access to the registry is available through the `roRegistry` object.

This object is created with no parameters.

```
CreateObject("roRegistry")
```

#### Interface: `ifRegistry`

The following methods are supported:

```
roList GetSectionList(Void)
```

- Returns a list with one entry for each registry section.

```
Boolean Delete(String section)
```

- Deletes the specified section and returns an indication of success.

```
Boolean Flush(Void)
```

- Flushes the registry out to persistent storage.

## 4.24 `roRegistrySection`

A Registry Section enables the organization of settings within the registry.

This object must be supplied with a "section" name on creation.

```
CreateObject("roRegistrySection", String section)
```

#### Interface: `ifRegistrySection`

The `roRegistrySection` object implements the `ifRegistrySection` interface. This interface provides:

```
String Read(String key)
```

- Reads and returns the value of the specified key.

```
Boolean Write(String key, String value)
```

- Replaces the value of the specified key.
- Does not guarantee a commit to non-volatile storage until an explicit `Flush()` is done.

```
Boolean Delete(String key)
```

- Deletes the specified key.

```
Boolean Exists(String key)
```

- Returns true if the specified key exists.

```
Boolean Flush(Void)
```

- Flushes the contents of the registry out to persistent storage. Developers should explicitly `Flush` after performing a write or series of writes. `Flush` is transactional and all writes between calls to `Flush` are atomic.

```
roList GetKeyList(Void)
```

- Returns a list containing one entry per registry key in this section.

#### Example: roRegistry/roRegistry Section usage

The following example shows two functions which get and set some user authentication in the registry.

```
Function GetAuthData() As Dynamic
 sec = CreateObject("roRegistrySection", "Authentication")
 if sec.Exists("UserRegistrationToken")
 return sec.Read("UserRegistrationToken")
 endif
 return invalid
End Function

Function SetAuthData(userToken As String) As Void
 sec = CreateObject("roRegistrySection", "Authentication")
 sec.Write("UserRegistrationToken ", userToken)
 sec.Flush()
End Function
```

## 4.25 roUrlTransfer

The URL Transfer object is used for reading from and writing to remote servers through URLs. It can perform mutual authentication with your web server.

This object is created with no parameters:

- `CreateObject("roUrlTransfer")`

The URL Transfer object authenticates your web server by calling `SetCertificatesFile()` with a .pem file that includes the certificate authority cert for the authority (like Verisign, Thawte, etc... or your own with OpenSSL) that signed your web server certificate before making the https request.

Your web server can authenticate that the requested connection is from a Roku Streaming Player and that the request is from your application by taking the following actions:

- 1) Add the Roku CA certificate to the web server's certificate authorities keychain. The Roku CA certificate is available in the SDK distribution package, in `certs/cacert.pem`
- 2) Configure your web server to reject any connection that does not have a valid client certificate.
- 3) Check the X-Roku-Reserved-Dev-Id header in the request. It should contain the Developer ID of your application. If it does not, another application on the Roku is attempting to access the server, so the request should be rejected.

In order for your web server to perform the steps above to authenticate your Roku Streaming Player, your application needs to call the following functions before performing any https requests:

```
object.AddHeader("X-Roku-Reserved-Dev-Id", "")
object.InitClientCertificates()
```

#### Interface: ifUrlTransfer

The interface provides:

```
Integer GetIdentity(Void)
 ▪ Returns a magic number that can be used to identify whether events originated from
 this object.

Void SetUrl(String URL)
 ▪ Sets the URL to use for the transfer request.

Void SetPort(ifMessagePort port)
 ▪ Set the message port to which events will be posted for asynchronous requests.

Void AddHeader(String name, String value)
 ▪ Add the specified HTTP header. Only valid for HTTP URLs.
 ▪ If "x-roku-reserved-dev-id" is passed as name, the Roku box ignores the passed in
 value. In its place, sends the devid of the application publisher of the currently
 running app. This allows the developers server to know the client app talking to it is
 on the Roku box.

Boolean SetHeaders(roAssociativeArray nameValueMap)
 ▪ Add the specified HTTP headers in the nameValueMap associative array.
 ▪ Header limitations specified in AddHeader() still apply.

String GetToString(Void)
 ▪ Connect to the remote service as specified in the URL and return the response body
 as a string. This function cannot return until the exchange is complete and it may
 block for a long time.
 ▪ Only having a single string return means that much of the information (headers,
 response codes) is discarded. If this information is required then use
 AsyncGetToString instead.

Integer GetToFile(String filename)
 ▪ Connect to the remote service as specified in the URL and write the response body
 to the specified file.
 ▪ This function does not return until the exchange is complete and may block for a long
 time.
 ▪ The response code from the server is returned. It is not possible to access any of the
 response headers. If this information is required use AsyncGetToFile instead.

Boolean AsyncGetToString(Void)
 ▪ Begin a get request to a string asynchronously. Events will be sent to the message
 port associated with the object. If false is returned then the request could not be
 issued and no events will be delivered.

Boolean AsyncGetToFile(String filename)
 ▪ Begin a get request to a file asynchronously. Events will be sent to the message port
 associated with the object. If false is returned then the request could not be issued
 and no events will be delivered.

roUrlEvent Head(Void)
 ▪ Synchronously perform an HTTP HEAD request and return the resulting response
 code and headers through a roUrlEvent object. In the event of catastrophic failure
 (e.g. an asynchronous operation is already active) then a null object is returned.

Boolean AsyncHead(Void)
 ▪ Begin an HTTP HEAD request asynchronously. Events will be sent to the message
 port associated with the object. If false is returned then the request could not be
 issued and no events will be delivered.

Integer PostFromString(String request)
 ▪ Use the HTTP POST method to post the supplied string to the current URL and
 return the response code. Any response body is discarded.

Integer PostFromFile(String filename)
 ▪ Use the HTTP POST method to post the contents of the file specified to the current
 URL and return the response code. Any response body is discarded.

Boolean AsyncPostFromString(String request)
```

- Use the HTTP POST method to post the supplied string to the current URL. Events of type `roUrlEvent` will be sent to the message port associated with the object. If false is returned then the request could not be issued and no events will be delivered.

```

Boolean AsyncPostFromFile(String filename)
 ▪ Use the HTTP POST method to post the contents of the specified file to the current
 URL. Events of type roUrlEvent will be sent to the message port associated with the
 object. If false is returned then the request could not be issued and no events will be
 delivered.
Boolean AsyncCancel(Void)
 ▪ Cancel any outstanding async requests on the roUrlTransfer object.
Boolean SetUserAndPassword(String user, String password)
 ▪ Enables HTTP authentication using the specified user name and password. Note that
 HTTP basic authentication is deliberately disabled due to it being inherently insecure.
 HTTP digest authentication is supported.

Boolean SetMinimumTransferRate(Integer bytes_per_second,
 Integer period_in_seconds)
 ▪ Cause the transfer to be terminated if the rate drops below bytes_per_second when
 averaged over period_in_seconds. Note that if the transfer is over the Internet you
 may not want to set period_in_seconds to a small number in case network problems
 cause temporary drops in performance. For large file transfers and a
 small bytes_per_second limit averaging over fifteen minutes or even longer might be
 appropriate.
String GetFailureReason(Void)
 ▪ If any of the roUrlTransfer functions indicate failure then this function may provide
 more information regarding the failure.
Boolean SetCertificatesFile(String path)
 ▪ Set the certificates file used for SSL to the .pem file specified. The .pem file should
 include the ca (certificate authority) cert that signed the cert installed on your web
 server. This enables authentication of your server.
 ▪ Instances of roUrlTransfer and ifHttpAgent components should call this function
 before performing https requests.
 ▪ The appropriate certificates file should be placed at the location specified in the
 SetCertificatesFile() function call.
Boolean InitClientCertificates(Void)
 ▪ Initialize the object to send the Roku client cert.
Boolean EnableEncodings(Boolean enable)
 ▪ Enable gzip encoding of transfers.
String Escape(String text)
 ▪ URL encode the specified string using CURL.
String Unescape(String text)
 ▪ Decode the specified string using CURL.
String UrlEncode(String url)
 ▪ Alternate URL encoding scheme with stricter RFC 1738 compliance.
String GetUrl(Void)
 ▪ Returns the URL currently set for this roUrlTransfer object.
Boolean EnableResume(Boolean enable)
 ▪ Enable automatic resumption of AsyncgetToFile and GetToFile requests.
Boolean EnablePeerVerification(Boolean enable)
 ▪ Verify the certificate has a chain of trust up to a valid root certificate using.
 CURLOPT_SSL_VERIFYPEER.
Boolean EnableHostVerification(Boolean enable)
 ▪ Verify that the certificate belongs to the host we're talking to using
 CURLOPT_SSL_VERIFYHOST.
Boolean EnableFreshConnection(Boolean enable)
 ▪ Enable fresh connection using CURLOPT_FRESH_CONNECT.

```

▪

#### Interface: ifSetMessagePort

Void SetMessagePort(roMessagePort port)

- Set the message port to be used for all events from the screen.

#### Interface: ifGetMessagePort

Object GetMessagePort(Void)

- Returns the message port currently set for use by the screen.

#### Events: roUrlEvent

The roUrlTransfer object sends the roUrlEvent with the ifUrlEvent interface for handling the transfer:

#### Interface: ifUrlEvent

Integer GetInt(Void)

- Returns the type of event. The following event types are currently defined:
  - 1 – transfer complete
  - 2 – transfer started. Headers are available for suitable protocols. (Not currently implemented)

Integer GetResponseCode(Void)

- Returns the protocol response code associated with this event. For a successful HTTP request this will be the HTTP status code 200.
- For unexpected errors the return value is negative. There are lots of possible negative errors from the CURL library but it's often best just to look at the text version via GetFailureReason().

Here are some potential errors. Not all of them can be generated.

| Status | Name                         | Description                                                                                       |
|--------|------------------------------|---------------------------------------------------------------------------------------------------|
| -1     | CURLE_UNSUPPORTED_PROTOCOL   |                                                                                                   |
| -2     | CURLE_FAILED_INIT            |                                                                                                   |
| -3     | CURLE_URL_MALFORMAT          |                                                                                                   |
| -5     | CURLE_COULDNT_RESOLVE_PROXY  |                                                                                                   |
| -6     | CURLE_COULDNT_RESOLVE_HOST   |                                                                                                   |
| -7     | CURLE_COULDNT_CONNECT        |                                                                                                   |
| -8     | CURLE_FTP_WEIRD_SERVER_REPLY |                                                                                                   |
| -9     | CURLE_REMOTE_ACCESS_DENIED   | a service was denied by the server due to lack of access - when login fails this is not returned. |
| -11    | CURLE_FTP_WEIRD_PASS_REPLY   |                                                                                                   |
| -13    | CURLE_FTP_WEIRD_PASV_REPLY   |                                                                                                   |
| -14    | CURLE_FTP_WEIRD_227_FORMAT   |                                                                                                   |

|     |                                |                                                        |
|-----|--------------------------------|--------------------------------------------------------|
| -15 | CURLE_FTP_CANT_GET_HOST        |                                                        |
| -17 | CURLE_FTP_COULDNT_SET_TYPE     |                                                        |
| -18 | CURLE_PARTIAL_FILE             |                                                        |
| -19 | CURLE_FTP_COULDNT_RETR_FILE    |                                                        |
| -21 | CURLE_QUOTE_ERROR              | quote command failure                                  |
| -22 | CURLE_HTTP_RETURNED_ERROR      |                                                        |
| -23 | CURLE_WRITE_ERROR              |                                                        |
| -25 | CURLE_UPLOAD_FAILED            | failed upload "command"                                |
| -26 | CURLE_READ_ERROR               | could open/read from file                              |
| -27 | CURLE_OUT_OF_MEMORY            |                                                        |
| -28 | CURLE_OPERATION_TIMEDOUT       | the timeout time was reached                           |
| -30 | CURLE_FTP_PORT_FAILED          | FTP PORT operation failed                              |
| -31 | CURLE_FTP_COULDNT_USE_REST     | the REST command failed                                |
| -33 | CURLE_RANGE_ERROR              | RANGE "command" didn't work                            |
| -34 | CURLE_HTTP_POST_ERROR          |                                                        |
| -35 | CURLE_SSL_CONNECT_ERROR        | wrong when connecting with SSL                         |
| -36 | CURLE_BAD_DOWNLOAD_RESUME      | couldn't resume download                               |
| -37 | CURLE_FILE_COULDNT_READ_FILE   |                                                        |
| -38 | CURLE_LDAP_CANNOT_BIND         |                                                        |
| -39 | CURLE_LDAP_SEARCH_FAILED       |                                                        |
| -41 | CURLE_FUNCTION_NOT_FOUND       |                                                        |
| -42 | CURLE_ABORTED_BY_CALLBACK      |                                                        |
| -43 | CURLE_BAD_FUNCTION_ARGUMENT    |                                                        |
| -45 | CURLE_INTERFACE_FAILED         | CURLOPT_INTERFACE failed                               |
| -47 | CURLE_TOO_MANY_REDIRECTS       | catch endless re-direct loops                          |
| -48 | CURLE_UNKNOWN_TELNET_OPTION    | User specified an unknown option                       |
| -49 | CURLE_TELNET_OPTION_SYNTAX     | Malformed telnet option                                |
| -51 | CURLE_PEER_FAILED_VERIFICATION | peer's certificate or fingerprint wasn't verified fine |
| -52 | CURLE_GOT_NOHING               | when this is a specific error                          |
| -53 | CURLE_SSL_ENGINE_NOTFOUND      | SSL crypto engine not found                            |
| -54 | CURLE_SSL_ENGINE_SETFAILED     | can not set SSL crypto engine as default               |
| -55 | CURLE_SEND_ERROR,              | failed sending network data                            |
| -56 | CURLE_RECV_ERROR               | failure in receiving network data                      |
| -58 | CURLE_SSL_CERTPROBLEM          | problem with the local certificate                     |
| -59 | CURLE_SSL_CIPHER               | couldn't use specified cipher                          |
| -60 | CURLE_SSL_CACERT               | problem with the CA cert (path?)                       |
| -61 | CURLE_BAD_CONTENT_ENCODING     | Unrecognized transfer encoding                         |
| -62 | CURLE_LDAP_INVALID_URL         | Invalid LDAP URL                                       |



|     |                             |                                                                                                                                                                                    |
|-----|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -63 | CURLE_FILESIZE_EXCEEDED,    | Maximum file size exceeded                                                                                                                                                         |
| -64 | CURLE_USE_SSL_FAILED,       | Requested FTP SSL level failed                                                                                                                                                     |
| -65 | CURLE_SEND_FAIL_REWIND,     | Sending the data requires a rewind that failed                                                                                                                                     |
| -66 | CURLE_SSL_ENGINE_INITFAILED | failed to initialise ENGINE                                                                                                                                                        |
| -67 | CURLE_LOGIN_DENIED          | user, password or similar was not accepted and we failed to login                                                                                                                  |
| -68 | CURLE_TFTP_NOTFOUND         | file not found on server                                                                                                                                                           |
| -69 | CURLE_TFTP_PERM             | permission problem on server                                                                                                                                                       |
| -70 | CURLE_REMOTE_DISK_FULL      | out of disk space on server                                                                                                                                                        |
| -71 | CURLE_TFTP_ILLEGAL          | Illegal TFTP operation                                                                                                                                                             |
| -72 | CURLE_TFTP_UNKNOWNID        | Unknown transfer ID                                                                                                                                                                |
| -73 | CURLE_REMOTE_FILE_EXISTS    | File already exists                                                                                                                                                                |
| -74 | CURLE_TFTP_NOSUCHUSER       | No such user                                                                                                                                                                       |
| -75 | CURLE_CONV_FAILED           | conversion failed                                                                                                                                                                  |
| -76 | CURLE_CONV_REQD             | caller must register conversion callbacks using curl_easy_setopt options CURLOPT_CONV_FROM_NETWORK_FUNCTION, CURLOPT_CONV_TO_NETWORK_FUNCTION, and CURLOPT_CONV_FROM_UTF8_FUNCTION |
| -77 | CURLE_SSL_CACERT_BADFILE    | could not load CACERT file, missing or wrong format                                                                                                                                |
| -78 | CURLE_REMOTE_FILE_NOT_FOUND | remote file not found                                                                                                                                                              |
| -79 | CURLE_SSH                   | error from the SSH layer, somewhat generic so the error message will be of interest when this has happened                                                                         |
| -80 | CURLE_SSL_SHUTDOWN_FAILED   | Failed to shut down the SSL connection                                                                                                                                             |

String GetFailureReason(Void)

- Returns a description of the failure that occurred.

String GetString(Void)

- Return the string associated with the event. For transfer complete AsyncGetToString, AsyncPostFromString and AsyncPostFromFile requests this will be the actual response body from the server truncated to 65536 characters.

Integer GetSourceIdentity(Void)

- Returns a magic number that can be matched with the value returned by roUrlTransfer.GetIdentity() to determine where this event came from.

roAssociativeArray GetResponseHeaders(Void)

- Returns an associative array containing all the headers returned by the server for appropriate protocols (such as HTTP).
- Headers are only returned when the status code is greater than or equal to 200 and less than 300.

#### Since Firmware version 2.6:

String GetTargetIpAddress(Void)

- Returns the ip address of the destination.

#### Since Firmware version 2.7:

Object GetResponseHeadersArray(Void)

- Returns an roArray of roAssociativeArrays. Each associative array contains a single header name/value pair.
- Use this function if you need access to duplicate headers as `GetResponseHeaders()` returns only the last name/value pair for a given name.
- All headers are returned regardless of the status code

## 5.0 Components First Introduced in Firmware v2.6

The following section provides details for all the platform specific components introduced in firmware version 2.6.

### 5.1 rolImageCanvas

The rolImageCanvas component provides an interface to render images at specific spots on the screen. Although it is not intended to be a full-fledged graphics component for high-performance gaming, it does provide a simple interface for building custom animations out of arrays of images displayed on the screen.

#### Interface: ifImageCanvas

The rolImageCanvas object implements the ifImageCanvas interface. This interface provides:

```
Void AllowUpdates(Boolean updatesEnabled)
 ▪ Turn updates on or off. Surrounding changes to several layers with AllowUpdates(false) and AllowUpdates(true) calls makes complex display modification atomic. Gives the application protection against the image canvas trying to render a partial update.
Void SetLayer(int zOrder, roAssociativeArray contentMetaData)
 ▪ zOrder is a z-order specifier with higher z-orders closer to the viewer. Negative z-orders are "behind the display" and are thus invisible. Each call to SetLayer replaces the previous contentMetaData that previously existed at that z-order layer. The caller passes an roAssociativeArrays (Content Meta-Data objects) representing the information for each image to be displayed on the rolImageCanvas. See "Content Meta-Data" for details on the attributes for each element in the array.
 ▪ The screen is redrawn when SetLayer() is called.
Void SetLayer(int zOrder, roArray contentList)
 ▪ Works like the previous SetLayer(), except passes an roArray of roAssociativeArrays instead of just a single roAssociativeArray of Content Meta-Data.
Void SwapLayers(int zOrderA, int zOrderB)
 ▪ Swap the content Meta-Data stored in zOrderA with the content Meta-Data stored in zOrderB.
 ▪ Enables the developer to change which layer is displayed "On Top"
 ▪ It is possible to SwapLayers with empty (unspecified) layers.
Void SetRequireAllImagesToDraw (Boolean requireAllImages)
 ▪ If true, wait till all images in the content-meta-data array are downloaded, decoded, and loaded into memory before drawing the screen.
 ▪ There is a large performance penalty for setting this to true.
Void PurgeCachedImages(Void)
 ▪ This is a convenience method that will purge the internal cache of all images related to urls in the current content list. If the content list is empty, then this method will do nothing.
roAssociativeArray GetCanvasRect(Void)
 ▪ Returns an roAssociativeArray with names w for width and h for height.
Void ClearLayer(Integer zOrder)
 ▪ Clear all content from a layer (see SetLayer for the layer definition)
```

`Void Clear(Void)`

- Clear all content from all layers.

`Boolean Show(Void)`

- Display or refresh the screen after initial creation or state changes.
- Note that screen update times can vary significantly depending on how much of the screen needs to be redrawn, and how many different images and layers you are using.

`Void Close(Void)`

- Close the screen and delete the associated object. Useful for avoiding screen flicker when the display order of your screens does not resemble a stack.

`Boolean AddButton(Integer id, String title)`

- Adds a button to the screen identified by the title and ID provided. The buttons are displayed in a standard location on the screen and appear in the order added. The ID is defined by the developer and used to uniquely identify the button instance. When the button is pressed, the script will receive an event from the application containing the ID of the button pressed and allowing the script to perform the desired action for that case.

#### Since Firmware version 2.7:

`Void ClearButtons(Void)`

- Clears all of the buttons from the screen and resets the array of buttons back to default with no buttons set.

#### Interface: `ifHttpAgent`

The `ifHttpAgent` methods setup the way URLs in the Content Meta-Data are accessed via this object.

`Boolean AddHeader(String name, String value)`

- Add the specified HTTP header.
- If “x-roku-reserved-dev-id” is passed as name, the Roku box ignores the passed in value. In its place, sends the devid of the application publisher of the currently running app. This allows the developer’s server to know which client app is talking to it.
- Developers may set any headers except what’s explicitly reserved by Roku. The Roku box will enforce ownership of values in the namespace x-roku-reserved-
- The Roku box will set initial values of well known headers (e.g. User-Agent, Content-Length, etc.)
- Users may override well known values if needed (e.g. some servers may require a specific user agent string)
- The Roku box will validate the data set in the headers to ensure it’s of the right type, not too long, etc.

`Boolean SetHeaders(roAssociativeArrayString nameValueMap)`

- Add the specified HTTP headers in the `nameValueMap` associative array.
- Header limitations specified in `AddHeader()` still apply.

`Boolean SetCertificatesFile(String path)`

- Set the certificates file used for SSL to the .pem file specified. The .pem file should include the ca (certificate authority) cert that signed the cert installed on your web server. This enables authentication of your server.
- Instances of `roUrlTransfer` and `ifHttpAgent` components should call this function before performing https requests.
- The appropriate certificates file should be placed at the location specified in the `SetCertificatesFile()` function call.

`Boolean InitClientCertificates(Void)`

- Initialize the object to send the Roku client cert.

### Example Code: ifHttpAgent

```
ba = CreateObject("roByteArray")
ba.FromAsciiString("myusername:mypassword")
myhttpagent.AddHeader("Authorization", "Basic " + ba.ToBase64String())
```

### Interface: ifSetMessagePort

```
Void SetMessagePort(roMessagePort port)
Set the message port to be used for all events from the screen.
```

### Interface: ifGetMessagePort

```
Object GetMessagePort(Void)
Returns the message port currently set for use by the screen.
```

### Events: rolImageCanvasEvent

The rolImageCanvas sends the rolImageCanvasEvent with the following predicates that indicate its valid event types:

```
isScreenClosed()
 ○ Event indicating that the screen was closed and no longer displayed to the user.
 ○ Event Details: Type = 1, Msg = "", Index = 0, Data = 0.

isRemoteKeyPressed()
 ○ Event indicating the user pressed a key on their remote control.
 ○ Event Details: Type = 7, Msg = "", Index = Integer ID of key pressed, Data = 0
```

### Example: rolImageCanvas

The following example displays two images and some text on the screen.

```
Sub showImageCanvas()

 canvasItems = [{ url:"http://192.168.1.23/boardwalk.jpg"
 TargetRect:{x:100,y:100,w:400,h:300}
 },
 { url:"http://192.168.1.23/walking.jpg"
 TargetRect:{x:500,y:400,w:400,h:300}
 },
 { Text:"Hello ImageCanvas"
 TextAttrs:{Color:"#FFCCCC", Font:"Medium",
 HAlign:"HCenter", VAlign:"VCenter",
 Direction:"LeftToRight"}
 TargetRect:{x:390,y:357,w:500,h:60}
 },
]

 canvas = CreateObject("roImageCanvas")
```

```

port = CreateObject("roMessagePort")
canvas.SetMessagePort(port)
'Set opaque background
canvas.SetLayer(0, {Color:"#FF000000", CompositionMode:"Source"})

canvas.SetRequireAllImagesToDraw(true)
canvas.SetLayer(1, canvasItems)
canvas.Show()

while(true)
 msg = wait(0,port)
 if type(msg) = "roImageCanvasEvent" then
 if (msg.isRemoteKeyPressed()) then
 i = msg.GetIndex()
 print "Key Pressed - " ; msg.GetIndex()
 if (i = 2) then
 ' Up - Close the screen.
 canvas.close()
 end if
 else if (msg.isScreenClosed()) then
 print "Closed"
 return
 end if
 end if
end while
End Sub

```

## 5.2 roFontRegistry

The roFontRegistry object allows you to register TrueType and OpenType fonts packaged with your application. Font files can quickly get very large, so be conscious of the size of the font files you include with your application. You should be able to find very good font files that are 50k or less. Anything larger is probably too big. The customvideoplayer sample application is a good example of usage. The font is available only for as long as the roFontRegistry object that registered it is not deleted.

### Interface: ifFontRegistry

The roFontRegistry object implements the ifFontRegistry interface. This interface provides:

```

Boolean Register(String path)
 ▪ Register a font file (.ttf or .otf) as an available font for your rolmageCanvas "Font" content meta-data parameter.
 ▪ Path is a valid path name (see Section 3.1)
roArray GetFamilies(Void)
 ▪ Returns an array of strings that represent the font family names for all font files registered with the roFontRegistry
String Get(String Family, integer size, integer weight, Boolean italic)
 ▪ Returns a valid font string that can be used as the value of the Font content meta-data parameter recognized by the rolmageCanvas.
 ▪ Family is a registered family name from .ttf or .otf files registered with this roFontRegistry
 ▪ Size, weight, italic are additional font characteristics that the .ttf or .otf files support.

```

## 5.3 roFontMetrics

The roFontMetrics object allows you to get display size information for a specific font returned by the roFontRegistry.Get() method. First, you need to initialize the roFontMetrics object with a font name that had been previously registered with the roFontRegistry, then the total rendered size of strings in that font can be returned by roFontMetrics.Size().

This object is created with a string that represents the font to use in its size calculations:

- CreateObject("roFontMetrics", String font)

### Interface: ifFontMetrics

The roFontMetrics object implements the ifFontMetrics interface. This interface provides:

roAssociativeArray Size(String stringToDraw)

- Returns an roAssociative array with parameters w and h set to the height and width of the stringToDraw rendered in the font passed on the CreateObject() call.
- Example return: { w:256, h:72 }

### Example: roFontMetrics

The example here shows the simple use of roFontRegistry and roFontMetrics to render a string on the roImageCanvas

```
helloString = "Hello ImageCanvas"

fontReg = CreateObject("roFontRegistry")
fontReg.Register("pkg:/fonts/LCDMono.ttf")
font = fontReg.Get("LCDMono",36,50,false) ' 36pt, 50 is normal
 ' weight, no italics

fontMetrics = CreateObject("roFontMetrics", font)

stringSize = fontMetrics.size(helloString)

canvasItem = { Text:helloString
 TextAttrs:{Color:"#FFCCCC", Font:font,
 HAlign:"HCenter",
 VAlign:"VCenter",
 Direction:"LeftToRight"}
 TargetRect:{x:390,y:357,
 w:stringSize.w,h:stringSize.h}
 }

canvas = CreateObject("roImageCanvas")
port = CreateObject("roMessagePort")
canvas.SetMessagePort(m.port)
'Set opaque background
canvas.SetLayer(0, {Color:"#FF000000", CompositionMode:"Source"})

canvas.SetRequireAllImagesToDraw(true)
canvas.SetLayer(1, canvasItem)
```

```
canvas.Show()
```

## 5.4 roSystemLog

The roSystemLog component enables the application to receive events from the Roku Streaming Player that are intended for reporting errors and trends, rather than trigger a response to a user action. The roSystemLog component requires specific Design Patterns in your BrightScript Application. Take care to:

- Use one roMessagePort throughout the application (instead of creating a new roMessagePort for each screen)
- Create one roSystemLog instance at startup that remains for the entire lifetime of the application.
- Pass the global roMessagePort referenced in the first bullet point to SetMessagePort() on the roSystemLog component.
- Enable the desired log types using EnableType()
- Handle the roSystemLogEvents in all message loops.

roSystemLogEvents implement the common event interface (GetIndex(), GetInfo(), etc.) however only GetInfo() returns usable information. roSystemLogEvents have an Info parameter named "LogType" that identifies the message. The rest of the data in the associative array is dependent on the LogType. All of the log events are sent to the roMessagePort that is registered on the roSystemLog object.

This object is created with no parameters:

- `CreateObject("roSystemLog")`

The roSystemLog object implements the following interfaces:

### Interface: ifSystemLog

```
Void EnableType(String logType)
```

- Enables log message of type: logType. When a logType is enabled the log events are sent to the message port set using the SetMessagePort() method. See the event section below for the valid values for logType. All system log events are disabled by default and must be explicitly enabled by the application. The current valid logTypes are:
  - **"http.error"** – Sent whenever an error occurs while executing an HTTP request. This could be at the time of the initial connection or after the initial connection (while reading data from the body).
  - **"http.connect"** – Sent whenever a successful http "connection" is made. This is a little unintuitive. It means that the server responded to the http request and there was no error in the headers of the response. This doesn't imply anything about the body of the http request because we could be reading that for a long time.
  - **bandwidth.minute** – bandwidth log events are sent every minute to indicate the current measured bandwidth.

### Interface: ifHttpAgent

The ifHttpAgent methods setup the way URLs in the Content Meta-Data are accessed via this object.

```
Boolean AddHeader(String name, String value)
```

- Add the specified HTTP header.

- If “x-roku-reserved-dev-id” is passed as name, the Roku box ignores the passed in value. In its place, sends the devid of the application publisher of the currently running app. This allows the developer’s server to know which client app is talking to it.
- Developers may set any headers except what’s explicitly reserved by Roku. The Roku box will enforce ownership of values in the namespace x-roku-reserved-
- The Roku box will set initial values of well known headers (e.g. User-Agent, Content-Length, etc.)
- Users may override well known values if needed (e.g. some servers may require a specific user agent string)
- The Roku box will validate the data set in the headers to ensure it’s of the right type, not too long, etc.

`Boolean SetHeaders(roAssociativeArrayString nameValueMap)`

- Add the specified HTTP headers in the nameValueMap associative array.
- Header limitations specified in AddHeader() still apply.

`Boolean SetCertificatesFile(String path)`

- Set the certificates file used for SSL to the .pem file specified. The .pem file should include the ca (certificate authority) cert that signed the cert installed on your web server. This enables authentication of your server.
- Instances of roUrlTransfer and ifHttpAgent components should call this function before performing https requests.
- The appropriate certificates file should be placed at the location specified in the SetCertificatesFile() function call.

`Boolean InitClientCertificates(Void)`

- Initialize the object to send the Roku client cert.

#### Example Code: ifHttpAgent

```
ba = CreateObject("roByteArray")
ba.FromAsciiString("myusername:mypassword")
myhttpagent.AddHeader("Authorization", "Basic " + ba.ToBase64String())
```

#### Interface: ifSetMessagePort

`Void SetMessagePort(roMessagePort port)`

Set the message port to be used for all events from the component.

#### Interface: ifGetMessagePort

`Object GetMessagePort(Void)`

Returns the message port currently set for use by the component.

#### Events: roSystemLogEvent

roSystemLogEvents have a type that is indicated by the value of the “LogType” key in the roAssociativeArray returned by GetInfo(). All roSystemLogEvents have the following keys:

- LogType - Identifies specific roSystemLogEvent.
- Datetime – an roDateTime object representing the time of the event with a resolution of one second. This, like all roDateTime objects, is GMT.
- Additional keys in the associative array are specific to each “LogType” event and are documented below:
  - **http.error** – No additional keys.



○ **http.connect**

| Key      | Value (String)                                                                                                                                                                                                                                                                                                                                                                                               |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Url      | The URL that was requested.                                                                                                                                                                                                                                                                                                                                                                                  |
| OrigUrl  | The original URL. If the original URL was redirected, then Url represents the new redirected URL and OrigURL the original. OrigURL is included so that it's easy to correlate between events and to urls passed to components.                                                                                                                                                                               |
| Method   | The HTTP method. "GET", "POST", or "HEAD"                                                                                                                                                                                                                                                                                                                                                                    |
| Status   | One of: <ul style="list-style-type: none"> <li>• ok (if and only if type = http.connect)</li> <li>• unknownerror</li> <li>• dnsfailure</li> <li>• dnstimeout</li> <li>• noroutetohost</li> <li>• connectiontimeout</li> <li>• connectionrefused</li> <li>• untrustedcert</li> <li>• expiredcert</li> <li>• nocipher</li> <li>• handshakefailed</li> <li>• generalsocketerror</li> <li>• httperror</li> </ul> |
| TargetIp | The ip address of the target server.                                                                                                                                                                                                                                                                                                                                                                         |
| HttpCode | The HTTP response code if available.                                                                                                                                                                                                                                                                                                                                                                         |

○ **bandwidth.minute**

| Key       | Value (String)              |
|-----------|-----------------------------|
| Bandwidth | Measured bandwidth in kbps. |

**Example: roSystemLog**

```

Sub showVideoScreenWithLogging(item As Object)
 port = CreateObject("roMessagePort")
 vs = CreateObject("roVideoScreen")
 ' **** Metrics ****
 ' Create a SystemLog object for detailed HTTP information and
 ' periodic bandwidth measurements. The high level idea is to use
 ' the roVideoScreen events as the primary driver of the
 ' reporting and then to use the http.error and http.connect
 ' roSystemLogEvents for detailed information. In this case that
 ' detailed information is primarily the ip addresses.
 syslog = CreateObject("roSystemLog")
 syslog.SetMessagePort(port)
 syslog.EnableType("http.error")
 syslog.EnableType("http.connect")
 syslog.EnableType("bandwidth.minute")
 ' **** End Metrics ****

 vs.SetContent(item)
 vs.SetPostionNotificationPeriod(1)
 vs.SetMessagePort(port)
 vs.Show()

```

```

metrics = CreateObject("roAssociativeArray")
metrics.streamStartTimer = CreateObject("roTimespan")
metrics.timeSpentBuffering = 0
metrics.errorCount = 0
while true
 if msg.isPlayingbackPosition() then
 if metrics.streamStartTimer <> invalid
 duration = metrics.streamStartTimer.TotalMilliseconds()
 dateTime = CreateObject("roDateTime").asSeconds()*1000
 startTime = dateTime - duration
 note = "Rebuffering"
 if lastpos = 0 note="Initial loading"
 print "Report following prints via urls to your site"
 print "Note is " ; note
 print "Report startTime, buffering, duration, note"
 end if
 elseif type(msg) = "roSystemLogEvent" then
 ' Handle the roSystemLogEvents:
 i = msg.GetInfo()
 if i.LogType = "http.error" or i.LogType = "http.connect"
 if i.LogType = "http.error"
 metrics.errorCount = metrics.errorCount + 1
 print "http error: "; i.HttpCode; "URL: ";i.Url)
 end if
 url = i.OrigUrl
 if (not httpIpAddrs.DoesExist(url)) then
 httpIpAddrs[url] = CeateObject("roAssociativeArray")
 end if
 httpIpAddrs[url].AddReplace(i.TargetIp, "")
 else if i.LogType = "bandwidth.minute"
 metrics.bandwidth = i.Bandwidth
 end if
 endif
 REM more event handling
end while
end Sub

```

## 5.5 roRegex

The roRegex component provides the powerful regular expression processing of the PCRE library to Brightscript strings. Please see the PCRE documentation (<http://www.pcre.org/>) for complete coverage of the possible expressions this library can parse and match. In general, most Perl compatible regular expressions are supported.

This object is created with a string that represents the *matching-pattern* and a string to indicate flags that modify the behavior of the matching operation(s):

- CreateObject("roRegex", "[a-z]+", "i")
  - The match string "[a-z]+" in the example above, which matches all lowercase letters) can include most Perl compatible regular expressions documented in the PCRE documentation (<http://www.pcre.org/>)
  - Any combination of the following behavior flags ("i" in the example above which modifies to match upper and lowercase letters) is supported:
    - "i" Case insensitive match

- "m" Multiline mode. The start of line "^" and end of line "\$" constructs match immediately following or before any newline in the subject string as well as the very start and end of the string. Normally, just the start and end of the string would match.
- "s" Sets dot-all mode that includes newline in the "." regular expression. This modifier is equivalent to Perl's /s modifier.
- "x" Sets extended mode that ignores whitespace characters except when escaped or inside a character class. Characters between an unescaped # outside a character a character class and the next newline character, inclusive, are also ignored. This modifier is equivalent to Perl's /x modifier

## Interface: ifRegex

The roRegex object implements the ifRegex interface. This interface provides:

Boolean IsMatch(String str)

- Returns true if str matches the matching pattern.
- Example from Brightscript Debugger Interactive Shell:  

```
> r = CreateObject("roRegex", "cad", "i")
> ? r.IsMatch("AbraCadabra")
true
>
```

roArray Match(String str)

- Returns an roArray of matched substrings from str.
- The entire match is returned in array[0]. If there are no parenthetical substrings this is the only entry in the array.
- If the matching pattern contains parenthetical substrings, the relevant substrings are returned as an array of length n+1, where array[0] is again the entire match and each additional entry in the array is the match for the corresponding parenthetical expression.
- Example from Brightscript Debugger Interactive Shell:  

```
> r = CreateObject("roRegex", "(a|(z))(bc)", "")
> ? r.Match("abcd")
abc
a

bc
>
```

String Replace(String str, String subst)

- Replaces the first occurrence of a match to matching pattern in str with subst.
- The subst may contain numbered back-references to parenthetical substrings.
- Example from Brightscript Debugger Interactive Shell:  

```
> r = CreateObject("roRegex", "(\d+)\s+(\w+)", "")
> ? r.Replace("123 abc", "\2 \1")
abc 123
>
```

String ReplaceAll(String str, String subst)

- Similar to Replace() but performs a global search and replace.
- Example from Brightscript Debugger Interactive Shell:  

```
> r = CreateObject("roRegex", "a", "i")
> ? r.ReplaceAll("Abracadabra", "x")
xbrxcxdxbrx
>
```

roList Split(String str)

- Uses the matching pattern as a delimiter and splits the string on the delimiter boundaries.

- Returns an roList of Strings that were separated by the matching pattern in the original str
- Example from Brightscript Debugger Interactive Shell:

```
> r = CreateObject("roRegex", "\s+", "")
> ? r.Split("The cow jumped over the moon.")
The
cow
jumped
over
the
moon.
>
```

## 5.6 roPath

The roPath component provides developers an easy way to create valid paths. Valid paths are discussed in Section 3.1. The roPath component is a convenience class that implements ifString while providing additional validation and path inspection functionality.

This object is created with a string that represents the initial path:

- `CreateObject("roPath", "ext1:/vid")`

### Interface: ifPath

The roPath object implements the ifPath interface. This interface provides:

`Boolean Change(String path)`

- Modify or change the current path via the relative or absolute path passed as a string
- Returns true if the resulting path is valid.

`Boolean IsValid(Void)`

- Returns true if the current path is valid.

`roAssociativeArray Split(Void)`

- Returns an roAssociativeArray containing the following keys:
  - Basename: (String) the filename, without parent directories or extension
  - extension: (String) the filename extension
  - filename: (String) the filename, with extension, without parent directories
  - parent: (String) the parent directory, or empty if in a root directory
  - phy: (String) the PHY volume
- Example from Brightscript Debugger Interactive Shell:

```
> mypath = CreateObject("roPath", "pkg:/source/appMain.brs")
> ? myPath.Split()
parent: pkg:/source/
extension: .brs
phy: pkg:
basename: appMain
filename: appMain.brs
>
```

## 5.7 roFileSystem

The roFileSystem component implements common filesystem inspection and modification routines. All paths are matched case-insensitively, regardless of the case-sensitivity of the underlying filesystem. The supported character set is limited to only those characters supported by vfat filesystems (valid Windows characters). The usbplayer sample application is a good example of roFileSystem usage. USB devices with VFAT, NTFS, HFS, and HFS Plus filesystems are supported. The USB filesystems are currently mounted read only.

This object is created with no parameters:

- `CreateObject("roFileSystem")`

## Interface: ifFileSystem

The roFileSystem object implements the ifFileSystem interface. This interface provides:

```
roList GetVolumeList(Void)
 ▪ Returns an roList of Strings of available volumes.
 ▪ Volumes may be internal or external storage devices.
roList Find(String dirPath, String regEx)
 ▪ Returns an roList of Strings representing the directory listing of names in dirPath
 matching the regEx regular expression.
roList FindRecurse(String dirPath, String regEx)
 ▪ Returns an roList of Strings representing the recursive directory listing of path names in
 dirPath matching the regEx regular expression.
 ▪ The Strings returned in the roList are relative paths to the dirPath parameter
roList Match(String path, String pattern)
 ▪ Returns an roList of Strings representing relative path names that are in path and match
 the passed pattern.
 ▪ The passed pattern may contain wildcards.
Boolean Exists(String path)
 ▪ Returns true if the path exists.
roAssociativeArray Stat(String path)
 ▪ Returns an roAssociativeArray containing the following keys for the passed in path:
 o type: (String) Either the value "file" or "directory"
 o size: (Integer) Number of bytes in the file. Only relevant for type
 "file".
 o permissions: (String) the value "rw" for read/write or "r" for read
 only.
roList GetDirectoryListing(String path)
 ▪ Returns an roList of Strings representing names that are in path and match the passed
 pattern.
roAssociativeArray GetVolumeInfo(String path)
 ▪ Returns an roAssociativeArray containing information about the volume specified in path.
 The following keys are returned in the roAssociativeArray:
 o Blocksize : (Integer) The size of the filesystem blocks in bytes.
 o Blocks : (Integer) The number of blocks in the filesystem.
 o Freeblocks : (Integer) The number of unused blocks in the filesystem.
 o Usedblocks : (Integer) The number of used blocks in the filesystem.
 o Label : (String) The volume label.
 ▪ Can only be called on external volumes. Internal volumes do not return any information.
Boolean CreateDirectory(String path)
 ▪ Creates the directory specified by the path parameter.
 ▪ Returns true if successful.
Boolean Delete(String path)
 ▪ Recursively deletes all contents under the specified path.
 ▪ Returns true if successful.
Boolean CopyFile(String fromPath, String toPath)
 ▪ Copies the file fromPath to toPath.
 ▪ Returns true if successful.
Boolean Rename(String fromPath, String toPath)
 ▪ Renames/Moves file or directory fromPath to toPath.
 ▪ If toPath exists, it is not clobbered, the operation fails, and Rename returns false.
 ▪ Returns true if successful.
```

### Interface: ifSetMessagePort

Void SetMessagePort(roMessagePort port)

Set the message port to be used for all events from the component.

### Interface: ifGetMessagePort

Object GetMessagePort(Void)

Returns the message port currently set for use by the component.

### Events: roFileSystemEvent

The roFileSystem component sends the roFileSystemEvent with the following predicates that indicate its valid event types:

**isStorageDeviceAdded()**

- Event indicating that a storage device was inserted in the USB port.
- Event Details: Type = 18, Msg = "<Volume Name>", Index = 0, Data = 0

**isStorageDeviceRemoved()**

- Event indicating that a storage device was removed from the USB port.
- Event Details: Type = 19, Msg = "<Volume Name>", Index = 0, Data = 0

## 5.8 roVideoPlayer

The roVideoPlayer component implements a video player with more programmatic control, but less user control than the roVideoScreen component. The roVideoPlayer can be used in conjunction with the rolmageCanvas to do graphical overlays, windowed video, zoom, and programmatic control of playlists and trick play. When using with the rolmageCanvas, you can put the roVideoPlayer is at a lower z-order layer than other imageCanvas layers and implement overlays on top of the playing video.

Unlike the roVideoScreen component roVideoPlayer does not have automatic trick play modes and built in controls to support that trick play. Any trick play requires the developer to build his own controls using buttons on the rolmageCanvas.

Note that all the video playback notes under roVideoScreen apply to the roVideoPlayer.

The customvideoplayer sample application is a good example of roVideoPlayer usage.

This object is created with no parameters:

- CreateObject("roVideoPlayer")

### Interface: ifVideoPlayer

The roVideoPlayer object implements the ifVideoPlayer interface. This interface provides:

Void SetContentList(Array contentList)

- Set the content to be played by the roVideoPlayer. The caller passes an Array of AssociativeArrays (Content Meta-Data objects) representing the information for each

stream to be played. See “Content Meta-Data” for details on the attributes for each element in the array.

- roVideoPlayer playback buffers on each Content item transition.

Void AddContent(AssociativeArray contentItem)

- Add a new Content Meta-Data item to the content list for the roVideoPlayer. New items are added to the end of the list.
- roVideoPlayer playback buffers on each Content item transition.

Void ClearContent(Void)

- Clear all content from the roVideoPlayer.

Boolean Play(Void)

- Put the roVideoPlayer into *play* mode starting at the beginning of the content list.
- This will stop any currently playing Content List.

Boolean Stop(Void)

- Stop roVideoPlayer from playing or pausing and cleanup.

Boolean Pause(Void)

- Put roVideoPlayer into *pause* mode.
- It is an error to Pause if player is not in play mode.

Boolean Resume(Void)

- Put roVideoPlayer into *play* mode starting from the pause point.
- It is an error to Resume from any other mode than *pause*.

Void SetLoop(Boolean loop)

- Enable/Disable the automatic replaying of the content list.
- Buffers on every loop to the beginning of the content list.

Void SetNext(Integer item)

- Set what the next item to be played within the Content List should be.

Boolean Seek(Integer OffsetMs)

- Set the start point of playback for the current video to OffsetMs milliseconds.

Void SetPositionNotificationPeriod(Integer period)

- Set interval to receive playback position events from the roVideoPlayer. The notification period is specified in seconds. Notification events sent to the script specify the position in seconds relative to the beginning of the stream.

Void SetCGMS(Integer level)

- Set CGMS (Copy Guard Management System) on analog outputs to the desired level.
  - 0 - CGMS off
  - 1 - No Copy Restriction
  - 2 - Copy No More
  - 3 - Copy Once Allowed
  - 4 – No Copying Permitted

Void SetDestinationRect(roAssociativeArray rect)

- Set the target display window for the video.
- rect has the params: {x: Integer, y: Integer, w: Integer, h: Integer}
- Default value is: {x:0, y:0, w:0, h:0}, which is full screen



### Since Firmware version 4.1:

`Void SetMaxVideoDecodeResolution(Integer width, Integer height)`

- Set the max resolution required by your video.
- Video decode memory is a shared resource with OpenGL texture memory. The Brightscript 2D APIs are implemented using OpenGL texture memory on Roku models that support the Open GL APIs (please see Section 1.4 of the Developer Guide for a list of these models).
- On models that do not support Open GL APIs this method exists for API compatibility but has no effect on actual memory allocations.
- Video decode memory allocation is based on a resolution of 1920x1080 or 1080x720 as the maximum supported resolution for a particular Roku model (please see Section 1.4 of the Developer Guide for a list of these models).
- This API enables applications that want to use both the 2D APIs and video playback with a lower resolution than 1080p. Without this call, these applications are likely to not have enough memory for either video playback or roScreen rendering.

### Deprecated Since Firmware version 3.0:

`Void SetMacrovisionLevel(Integer level)`

- Deprecated stub function does nothing. Roku no longer supports Macrovision and this function exists as a no-op so that legacy scripts do not break.

### Interface: ifHttpAgent

The ifHttpAgent methods setup the way URLs in the Content Meta-Data are accessed via this object.

`Boolean AddHeader(String name, String value)`

- Add the specified HTTP header.
- If “x-roku-reserved-dev-id” is passed as name, the Roku box ignores the passed in value. In its place, sends the devid of the application publisher of the currently running app. This allows the developer’s server to know which client app is talking to it.
- Developers may set any headers except what’s explicitly reserved by Roku. The Roku box will enforce ownership of values in the namespace x-roku-reserved-
- The Roku box will set initial values of well known headers (e.g. User-Agent, Content-Length, etc.)
- Users may override well known values if needed (e.g. some servers may require a specific user agent string)
- The Roku box will validate the data set in the headers to ensure it’s of the right type, not too long, etc.

`Boolean SetHeaders(roAssociativeArrayString nameValueMap)`

- Add the specified HTTP headers in the nameValueMap associative array.
- Header limitations specified in AddHeader() still apply.

`Boolean SetCertificatesFile(String path)`

- Set the certificates file used for SSL to the .pem file specified. The .pem file should include the ca (certificate authority) cert that signed the cert installed on your web server. This enables authentication of your server.
- Instances of roUrlTransfer and ifHttpAgent components should call this function before performing https requests.
- The appropriate certificates file should be placed at the location specified in the SetCertificatesFile() function call.

`Boolean InitClientCertificates(Void)`

- Initialize the object to send the Roku client cert.

#### Example Code: ifHttpAgent

```
ba = CreateObject("roByteArray")
ba.FromAsciiString("myusername:mypassword")
myhttpagent.AddHeader("Authorization", "Basic " + ba.ToBase64String())
```

#### Interface: ifSetMessagePort

Void SetMessagePort(roMessagePort port)  
Set the message port to be used for all events from the component.

#### Interface: ifGetMessagePort

Object GetMessagePort(Void)  
Returns the message port currently set for use by the component.

#### Events: roVideoPlayerEvent

The roVideoPlayer sends the roVideoPlayerEvent with the following predicates that indicate its valid event types:

```
isScreenClosed()
 o Event indicating that the screen was closed and no longer displayed to the user.
 o Event Details: Type = 1, Msg = "", Index = 0, Data = 0
isPlaybackPosition()
 o Periodic event to the script to indicate the current position in the video stream.
 o Event Details: Type = 6, Msg = "", Index = position offset in seconds, Data = 0
isRequestFailed()
 o Event to indicate error in video playback
 o Event Details: Type = 9, Msg = <text description of error>, Index = <error id>,
 Data = 0
 ▪ Error number codes:
 • 0 = Network error : server down or unresponsive, server is
 unreachable, network setup problem on the client.
 • -1 = HTTP error: malformed headers or HTTP error result.
 • -2 = Connection timed out
 • -3 = Unknown error
 ▪ Info = { url:<url of video>,
 StreamBitrate:<bitrate of stream>,
 MeasuredBitrate:<actualBitrateOfDataTransfer>
 }
 ▪ Info was Introduced in firmware v2.6

isStatusMessage()
 o Event with status messages about playback.
 o Event Details: Type = 11, Msg = <video player defined messages>, Index = "",
 Index=0, Data = 0
isFullResult()
```

- Event indicating that playback completed at end of content
- Event Details: Event Details: Type = 16, Msg = "", Index = 0, Data = 0
- isPartialResult()**
  - Event to indicate playback was interrupted
  - Event Details: Type = 15, Msg = "", Index = 0, Data = 0
- isStreamStarted()**
  - Event to give details about the stream that was started.
  - Event Details: Type = 20, Msg = "", Index = Seconds from Play to Start Streaming, Data = 0,
    - Info = { url:<url of video>, StreamBitrate:<bitrate of stream in kbps>, MeasuredBitrate:<actualBitrateOfDataTransfer in kbps>, IsUnderrun:<true if streamStarted is the result of an underrun> }
- isPaused()**
  - Event indicating that video playback was paused by the user.
  - Event Details: Type = 12, Msg = "", Index = 0, Data = 0
- isResumed()**
  - Event indicating that video playback has resumed.
  - Event Details: Type = 13, Msg = "", Index = 0, Data = 0

## 6.0 Components First Introduced in Firmware v2.7

The following section provides details for all the platform specific components introduced in firmware version 2.7.

### 6.1 roGridScreen

The Grid Screen provides a graphical display of poster art from multiple content categories from within a single screen. Users can browse within a category list by scrolling horizontally and between category lists by scrolling vertically. There is an optional callout box in the lower right corner of the screen that can display details about the focused item without leaving the screen. Each item in the grid screen is represented by an image (aka poster), so any type of item that can be visually represented by an image can be displayed in the poster screen. It is used to show lists of data to users and common patterns include content categories, movies, podcasts, pictures, and search results.

The initial release of roGridScreen only enabled the default list style, “portrait”, using the following art sizes:

Artwork sizes: SD=110x150; HD=210x270

It also required grid posters to be .jpg files.

#### Since Firmware version 2.8:

File types of .png and .gif files are now supported, though they are converted internally to .jpg by the roGridScreen so they have a performance penalty.

In v2.8, there are now multiple grid styles that are specified in the `SetGridStyle()` method below. It's also worth going back and reviewing the appManager theme parameters in Section 4.2, as v2.8 adds some new grid parameters. The border around the focused poster screen can be customized with the GridScreenFocusBorder images in png format. PNG files can have a transparent color value that you will need to allow the focused poster image to show through the border image. The corresponding offsets should be negative offsets that would be up and to the left of the top left corner of the poster image. The width of the borders should be the absolute values of the offsets and the rest of the image should be transparent inside. The GridScreenDescriptionImage is also positioned relative to the top left corner of the focused image. It can be positioned up and to the left with negative x and y offsets, below and to the right with positive offsets, or in the other corners with mixed signed x and y offsets. It's recommended that you include a “callout” tip pointing to the focused image in the GridScreenDescriptionImage.

## Diagram: roGridScreen



This object is created with no parameters:

- `CreateObject("roGridScreen")`

The roGridScreen implements the following interfaces:

### Interface: ifGridScreen

- `Void SetupLists(Integer count)`
- Set the number of rows in the gridScreen.
- `Void SetListNames(Object names)`
- Sets the list of categories to be displayed as titles above each row of poster screens. The caller passes the list as an array of Strings. Each String represents a new category. The display order of rows is the same as the order of the categories in the array passed by the caller. In the screenshot above, "Instant Queue" and "Search Results for "24: Season 1" are the category names shown.
- `Void SetListName(Integer rowIndex, String name)`
- Sets the `rowIndex` member of the list of row names to the passed String name.
- `Void SetContentList(Integer rowIndex, Object contentList)`
- Set the list of content to be displayed in the `rowIndex` row of the grid. The caller passes the `rowIndex` and an `roArray` of `roAssociativeArrays` (Content Meta-Data objects) representing the information for each title to be displayed on screen. See "Content Meta-Data" for details on the attributes for each element in the array. The screen is responsible for fetching the poster art from the URL's specified and all user navigation within the list.
- `Void SetContentListSubset(Integer rowIndex, Object contentList, Integer offset, Integer length)`
- Enable a partial update of the GridScreen content lists. Enables performance improvements when dealing with your backend services by asking only for only a

screenful's worth of data at a time, and then update the rest of the content list content in the background.

- Parameters offset and length specify the sublist to fill of the row's content list to fill with the contents of the passed contentList array.
- Set the subset of the list of content to be displayed by in the rowIndex row of the grid. The caller passes the rowIndex and an roArray of roAssociativeArrays (Content Meta-Data objects) representing the information for each title to be displayed on screen. See "Content Meta-Data" for details on the attributes for each element in the array. The screen is responsible for fetching the poster art from the URL's specified and all user navigation within the list.

Void SetListOffset(Integer rowIndex, Integer itemIndex)

- The grid populates poster screens in a background thread.
- SetListOffset() enables a developer to change the current position (both row and column) of the downloading thread in populating image posters of the grid.
- The downloading thread will reprioritize poster downloads based on the new list offset.

Void SetListName(Integer rowIndex, String name)

- Sets the rowIndex member of the list of row names to the passed String name.

Void SetListVisible(Integer rowIndex, Boolean visible)

- If visible is true, display the row of listIndex when the user navigates to the row. If visible is false, do not let the user navigate to the row.

Void SetFocusedListItem(Integer listIndex, Integer itemIndex)

- Set the focus for the listIndex row. Zero-based index of item in poster list to be selected. The selected item is displayed in the center of the screen and bordered to designate focus.

Void SetDisplayMode(String displayMode)

- Sets the mode for displaying images in the grid screen. This allows images to be either scaled to completely fill the poster frame (scale-to-fill) or scaled to fit inside the poster frame (scale-to-fit) while maintaining aspect ratio. Valid display modes are:
  - scale-to-fill – scale image to completely fill the rectangle of the bounding frame (Default)
  - scale-to-fit. – scale image to fit horizontally or vertically as appropriate while still maintaining aspect ratio. Note that scale-to-fit may result in pillar-box or letter-box display of images.
  - zoom-to-fill – scales and crops image to maintain aspect ratio and completely fill the rectangle of the bounding frame.
  - photo-fit – Uses several methods to fit the image with a different aspect ratio to the screen. First, it will asymmetrically scale up to a maximum of 5%. Second, for landscape images, if vertical cropping is necessary, it will remove two lines off the bottom for every one line off the top up to a maximum of 30% of the image. For all images, if horizontal cropping is necessary it will crop an equal amount from both sides.

Void setDescriptionVisible(Boolean visible)

- When passed in visible Boolean is true, the description box is displayed. When it is false, the description box is not displayed. In the screenshot above, the description box is in the lower right corner and describes "30 Rock: Season 3".

Void ShowMessage(String message)

- Displays a semi-transparent popup message box to the user in the center of the screen over the poster screen. Generally used for error messages.

Void ClearMessage(Boolean clear)

- Clears the message from the previous ShowMessage call.

Boolean Show(Void)

- Display or refresh the screen after initial creation or state changes.

Void Close(Void)

- Close the screen and delete the associated object. Useful for avoiding screen flicker when the display order of your screens does not resemble a stack.

### Since Firmware version 2.8:

Void SetGridStyle(String style)

- Sets the style or theme for displaying images in the grid screen. This allows different appearances of the overall grid for different sized images:
  - flat-movie – movie posters as seen in the Netflix channel (Default)
    - Image sizes: SD 110x150 HD 210x270
    - SD 5 posters across, by 2 rows
    - HD 5 posters across, by 2 rows
  - flat-portrait
    - Image sizes: SD 110x140 HD 210x300
    - SD 5 posters across, by 2 rows
    - HD 5 posters across, by 2 rows
  - flat-landscape
    - Image sizes: SD 140x94 HD 210x158
    - SD 4 posters across, by 3 rows
    - HD 5 posters across, by 3 rows
  - flat-square – note that SD has non-square pixels so the dimensions below appear as square on the screen
    - Image sizes: SD 96x86 HD 132x132
    - SD 6 posters across, by 3 rows
    - HD 7 posters across, by 3 rows
  - flat-16x9
    - Image sizes: SD 140x70 HD 210x118
    - SD 4 posters across, by 3 rows
    - HD 5 posters across, by 3 rows

Void SetBreadcrumbEnabled(Boolean enabled)

- If enabled is true, display the breadcrumb text specified with SetBreadcrumbText

Void SetBreadcrumbText(String Location1, Location2)

- If SetBreadcrumbEnabled() is true, display Location1 and Location2 in the right of the overhang

Void setDescriptionVisible(Boolean enabled)

- If enabled is true, display the description callout image (showing 30 Rock: Season 3 ...) in the screenshot above. Otherwise, hide it.

### Since Firmware version 2.9:

Void SetUpBehaviorAtTopRow(String behavior)

- behavior is a string that controls how the remote “up” key behaves when pressed with the top row selected. The default behavior, “stop”, is to stop scrolling but stay on the roGridScreen. Valid values:
  - stop
    - stop scrolling up, and stay on the roGridScreen
    - default behavior
  - exit
    - exit the roGridScreen

## Interface: ifHttpAgent

The ifHttpAgent methods setup the way URLs in the Content Meta-Data are accessed via this object.

Boolean AddHeader(String name, String value)

- Add the specified HTTP header.
- If "x-roku-reserved-dev-id" is passed as name, the Roku box ignores the passed in value. In its place, sends the devid of the application publisher of the currently running app. This allows the developer's server to know which client app is talking to it.
- Developers may set any headers except what's explicitly reserved by Roku. The Roku box will enforce ownership of values in the namespace x-roku-reserved-
- The Roku box will set initial values of well known headers (e.g. User-Agent, Content-Length, etc.)
- Users may override well known values if needed (e.g. some servers may require a specific user agent string)
- The Roku box will validate the data set in the headers to ensure it's of the right type, not too long, etc.

Boolean SetHeaders(roAssociativeArrayString nameValueMap)

- Add the specified HTTP headers in the nameValueMap associative array.
- Header limitations specified in AddHeader() still apply.

Boolean SetCertificatesFile(String path)

- Set the certificates file used for SSL to the .pem file specified. The .pem file should include the ca (certificate authority) cert that signed the cert installed on your web server. This enables authentication of your server.
- Instances of roUrlTransfer and ifHttpAgent components should call this function before performing https requests.
- The appropriate certificates file should be placed at the location specified in the SetCertificatesFile() function call.

Boolean InitClientCertificates(Void)

- Initialize the object to send the Roku client cert.

## Example Code: ifHttpAgent

```
ba = CreateObject("roByteArray")
ba.FromAsciiString("myusername:mypassword")
myhttpagent.AddHeader("Authorization", "Basic " + ba.ToBase64String())
```

## Interface: ifSetMessagePort

Void SetMessagePort(roMessagePort port)

- Set the message port to be used for all events from the screen.

## Interface: ifGetMessagePort

Object GetMessagePort(Void)

- Returns the message port currently set for use by the screen.

## Events: roGridScreenEvent



The roGridScreen sends the roGridEvent with the following predicates that indicate its valid event types:

- isScreenClosed()**
  - Event type indicating that the screen was closed and no longer displayed to the user.
  - Event Details: Type = 1, Msg = "", Index = 0, Data = 0.
- isListItemFocused()**
  - Event type indicating that a new content item in the poster screen has gained focus.
  - Event Details: Type = 4, Msg = "", Index = Index of currently focused list (row), Data = index of focused item (column).
- isListItemSelected()**
  - Event type indicating that a content item in the poster screen has been selected.
  - Event Details: Type = 0, Msg = "", Index = Index of currently focused list (row), Data = index of focused item (column).
- isRemoteKeyPressed()**
  - Event type indicating that a IR remote key was pressed
  - Event Details: Type = 7 Msg = "", Index =<IR key as an Integer>, Data = 0

#### Example Code: roGridScreen

```
Function Main()
 port = CreateObject("roMessagePort")
 grid = CreateObject("roGridScreen")
 grid.SetMessagePort(port)

 rowTitles = CreateObject("roArray", 10, true)
 for j = 0 to 10
 rowTitles.Push("[Row Title "+j.toStr()+"] ")
 end for
 grid.SetupLists(rowTitles.Count())
 grid.SetListNames(rowTitles)

 for j = 0 to 10
 list = CreateObject("roArray", 10, true)
 for i = 0 to 10
 o = CreateObject("roAssociativeArray")
 o.ContentType = "episode"
 o.Title = "[Title"+i.toStr+"]"
 o.ShortDescriptionLine1 = "[ShortDescriptionLine1]"
 o.ShortDescriptionLine2 = "[ShortDescriptionLine2]"
 o.Description = ""
 o.Description = "[Description] "
 o.Rating = "NR"
 o.StarRating = "75"
 o.ReleaseDate = "[<mm/dd/yyyy]"
 o.Length = 5400
 o.Actors = []
 o.Actors.Push("[Actor1]")
 o.Actors.Push("[Actor2]")
 o.Actors.Push("[Actor3]")
 o.Director = "[Director]"

 list.Push(o)
 end for
 end for
```

```

 end for
 grid.SetContentList(j, list)
 end for

 grid.Show()

 while true
 msg = wait(0, port)
 if type(msg) = "roGridScreenEvent" then
 if msg.isScreenClosed() then
 return -1
 elseif msg.isListItemFocused()
 print "Focused msg: ";msg.GetMessage();"row: ";msg.GetIndex();
 print " col: ";msg.GetData()
 elseif msg.isListItemSelected()
 print "Selected msg: ";msg.GetMessage();"row: ";msg.GetIndex();
 print " col: ";msg.GetData()
 endif
 endif
 endif
End Function

```

## 7.0 Components First Introduced in Firmware v2.9

The following section provides details for all the platform specific components introduced in firmware version 2.9.

### 7.1 roAudioMetadata

The roAudioMetadata component provides developers access to audio file metadata included in many audio files. This should enable some audiofiles to deliver the information needed to fill out an roSpringboard screen without passing the info in a separate xml feed. roAudioMetadata currently only works with local file Urls.

The roAudioMetadata requires the use of a dynamically loaded library that is not part of the initially booted image. Therefore, an entry must be added to the manifest of any applications that use the roAudioMetadata component so it can be loaded when the channel is launched. Here's the manifest entry:

```
requires_audiometadata=1
```

This object is created without any arguments:

- `CreateObject("roAudioMetadata")`

#### Interface: ifAudioMetadata

```
void SetUrl(url)
```

- Set the URL to the audio file. Only file URL's are initially supported.

```
Object GetTags()
```

- Returns an associative array that contains a simple set of tags that are common to most audio formats. This associative array contains:

| Name     | Type    | Notes                                                                           |
|----------|---------|---------------------------------------------------------------------------------|
| title    | String  |                                                                                 |
| artist   | String  | Returns the first artist found even though many titles have multiple artists.   |
| album    | String  |                                                                                 |
| composer | String  | Returns the first composer found even though many titles have multiple artists. |
| comment  | String  |                                                                                 |
| genre    | String  |                                                                                 |
| year     | Integer |                                                                                 |
| track    | Integer |                                                                                 |

```
Object GetAudioProperties()
```

- Returns an aa with a simple set of audio properties. These are values which may involve reading a larger portion of the file and thus may take longer to retrieve than the tags. The associative array contains:

| Name       | Type    | Notes                           |
|------------|---------|---------------------------------|
| length     | Integer | Seconds.                        |
| bitrate    | Integer | kpbs                            |
| samplerate | Integer | Example: 44100                  |
| channels   | Integer | Number of channels. Example: 2. |

```
Object GetCoverArt()
```

- Returns the covert art if available. Returns an aa with two entries: “bytes” and “type”. “bytes” is an roByteArray with the image data. “type” specifies the mime-type of image which is almost always either “image/jpeg” or “image/png”.
- Looks for the picture designated as the cover art if there is more than one picture in the file. If there is no FrontCover picture then the first picture is used.

#### Example Code: roAudioMetadata

REM printAA() is from generalUtils.brs in our sample apps  
 REM and used to print an associative Array

```
Sub SaveCoverArtFile(filename As String)
 meta = CreateObject("roAudioMetadata")
 meta.SetUrl(filename)
 print "----- GetTags() -----"
 tags = meta.GetTags()
 printAA(tags)
 print "----- GetAudioProperties() -----"
 properties = meta.GetAudioProperties()
 printAA(properties)
 print "----- GetCoverArt() -----"
 thumbnail = meta.GetCoverArt()
 if (thumbnail <> invalid) then
 if (thumbnail.bytes = invalid) then
 return
 end if

 imgtype = thumbnail.type
 image_ext=""
 if (imgtype = "image/jpeg" or imgtype = "jpg") then
 image_ext = "jpg"
 else if (imgtype = "image/png" or imgtype = "png") then
 image_ext = "png"
 else
 image_ext = "jpg"
 end if

 tmp_img = "tmp:/CoverArtImage" + "." + image_ext
 if (tmp_img <> invalid) then
 DeleteFile(tmp_img)
 end if
 thumbnail.bytes.Writefile(tmp_img)
 end if
End Sub
```

## 7.2 rolImageMetadata

The rolImageMetadata component provides developers access to image file metadata included in many .jpg EXIF headers. rolImageMetadata currently only works with local file Urls.

This object is created without any arguments:

- CreateObject("roImageMetadata")

#### Interface: ifImageMetadata

```
void SetUrl(String url)
```

- Set the URL to the image. Only file urls are initially supported.

```
Object GetMetadata()
```

- Returns an associative array with set of simple and common image metadata. This associative array includes:

| Name        | Type       | Notes                                                                  |
|-------------|------------|------------------------------------------------------------------------|
| width       | Integer    | Width of the image in pixels.                                          |
| height      | Integer    | Height of the image in pixels.                                         |
| orientation | String     | “portrait” or “landscape”                                              |
| datetime    | roDateTime | The creation time of the image such as the time a photo was taken.     |
| comment     | String     | User specified comment string. This is often referred to as a caption. |

```
Object GetThumbnail()
```

- Returns a thumbnail image if one is embedded in the image metadata. This will not generate a thumbnail if one doesn't already exist. Returns an aa with two entries: “bytes” and “type”. “bytes” is an roByteArray with the image data. “type” specifies the type of image which is most likely “image/jpeg” but could be something else like “image/png”.

```
Object GetRawExif()
```

- Returns an associative array with all of the raw EXIF metadata. See the EXIF section below for more details.

```
Object GetRawExifTag(Integer ifd, Integer tag)
```

- Returns the raw data for one Exif tag. Returns invalid if that tag does not exist. This is useful for direct access to a raw EXIF tag if you know exactly what tag you want.

## EXIF Background

Each EXIF tag represents one piece of metadata. Each tag is uniquely identified by a tag number and the IFD in which it was found. All the tags are grouped into a small set of IFDs (Image File Directory). The EXIF specification describes 5 IFDs:

| Number | Name             | Notes                                                                                                                                     |
|--------|------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| 0      | image            | Tags related to image structure and some additional basic information.                                                                    |
| 1      | thumbnail        | Tags related to the thumbnail image structure                                                                                             |
| 2      | exif             | Tags related to non-image structure data. I know this is an odd name. Usually stuff like ApertureValue that a digital camera would write. |
| 3      | gps              | Tags related to GPS.                                                                                                                      |
| 4      | interoperability |                                                                                                                                           |

The associative array returned by GetRawExif() function on the rolmageMetadata component is organized as a tree where the first level consists of the IFDs, the second level is the tag name, and the third (leaf) level is tag associative array.

Each tag associative array contains the following values:

| Name  | Type    | Notes                                                         |
|-------|---------|---------------------------------------------------------------|
| Tag   | Integer | The tag number.                                               |
| Value | String  | This is a string representation of the data regardless of the |

|  |  |                       |
|--|--|-----------------------|
|  |  | underlying data type. |
|--|--|-----------------------|

Tag values can be one of several types (string, bytes, rational, enum, etc.) We are using a that converts most values to a string. In the future or time permitting, we may add more values to the tag aa to give access to raw bytes.

The best way to illustrate how the EXIF data is accessed is through some concrete examples using the data in the reference section below. Assume that the associative array aa contains the results of the GetRawExif() function.

- To get the camera model: aa.image.model.value
- To get the gps latitude: aa.gps.GPSLatitude.value

## Reference

For reference here are all the fields from an image taken from my camera (with the addition of GPS attributes I added using Picasa).

The format is IFD#, Tag#[Tag Name], Data Format, Value. The Value is a string representation of data.

```

0, 272[Model], ASCII,Canon PowerShot SD700 IS
0, 274[Orientation], Short,top - left
0, 283[YResolution], Rational,180.00
0, 296[ResolutionUnit], Short,Inch
0, 305[Software], ASCII,Picasa 3.0
0, 306[DateTime], ASCII,2007:08:14 10:06:48
0, 531[YCbCrPositioning], Short,centered
0, 282[XResolution], Rational,72.00
1, 259[Compression], Short,JPEG compression
1, 282[XResolution], Rational,180.00
1, 283[YResolution], Rational,180.00
1, 296[ResolutionUnit], Short,Inch
2,33434[ExposureTime], Rational,1/320 sec.
2,33437[FNumber], Rational,f/2.8
2,36864[ExifVersion], Undefined,Exif Version 2.2
2,36867[DateTimeOriginal], ASCII,2007:08:14 10:06:48
2,36868[DateTimeDigitized], ASCII,2007:08:14 10:06:48
2,37121[ComponentsConfiguration], Undefined,Y Cb Cr -
2,37122[CompressedBitsPerPixel], Rational,5.00
2,37377[ShutterSpeedValue], SRational,8.31 EV (1/317 sec.)
2,37378[ApertureValue], Rational,2.97 EV (f/2.8)
2,37380[ExposureBiasValue], SRational,0.00 EV
2,37381[MaxApertureValue], Rational,2.97 EV (f/2.8)
2,37383[MeteringMode], Short,Pattern
2,37385[Flash], Short,Flash did not fire, auto mode
2,37386[FocalLength], Rational,5.8 mm
2,37500[MakerNote], Undefined,1838 bytes undefined data
2,37510[UserComment], Undefined,
2,40960[FlashPixVersion], Undefined,FlashPix Version 1.0
2,40961[ColorSpace], Short,sRGB
2,40962[PixelXDimension], Short,2816
2,40963[PixelYDimension], Short,2112
2,41486[FocalPlaneXResolution], Rational,12515.56
2,41487[FocalPlaneYResolution], Rational,12497.04
2,41488[FocalPlaneResolutionUnit], Short,Inch
2,41495[SensingMethod], Short,One-chip color area sensor
2,41728[FileSource], Undefined,DSC
2,41985[CustomRendered], Short,Normal process
2,41986[ExposureMode], Short,Auto exposure
2,41987[WhiteBalance], Short,Auto white balance
2,41988[DigitalZoomRatio], Rational,1.00
2,41990[SceneCaptureType], Short,Standard
2,42016[ImageUniqueID], ASCII,ba6ad9a9d88ffd9576ea6155afa6c6ef
3, 0[GPSVersionID], Byte,2.2.0.0
3, 1[GPSLatitudeRef], ASCII,N
3, 2[GPSLatitude], Rational,40.00, 12.00, 44.79
3, 3[GPSLongitudeRef], ASCII,W

```

```

3, 4[GPSLongitude], Rational,75.00, 37.00, 47.58
3, 5[GPSAltitudeRef], Byte,Sea level
4, 1[InteroperabilityIndex], ASCII,R98
4, 2[InteroperabilityVersion], Undefined,0100
4, 4097[RelatedImageWidth], Short,2816
4, 4098[RelatedImageLength], Short,2112

```

### Example Code: roImageMetadata

REM printAA() is from generalUtils.brs in our sample apps  
 REM and used to print an associative Array

```

Sub SaveExifImage(filename As String)
 meta = CreateObject("roImageMetadata")
 meta.SetUrl(filename)
 print "----- GetRawExif() -----"
 allexif = meta.GetRawExif()
 printAA(allexif)
 print "----- GetMetadata() -----"
 simple = meta.GetMetadata()
 printAA(simple)
 print "----- GetRawExifTag() -----"
 rawexiftag = meta.GetRawExifTag(2,36868)
 printAA(rawexiftag)
 print "----- GetThumbnail() -----"
 thumbnail = meta.GetThumbnail()
 if (thumbnail <> invalid) then
 if (thumbnail.bytes = invalid) then
 return
 end if

 imgtype = thumbnail.type
 image_ext=""
 if (imgtype = "image/jpeg" or imgtype = "jpg") then
 image_ext = "jpg"
 else if (imgtype = "image/png" or imgtype = "png") then
 image_ext = "png"
 else
 image_ext = "jpg"
 end if

 tmp_img = "tmp:/TmpExifImage" + "." + image_ext
 if (tmp_img <> invalid) then
 DeleteFile(tmp_img)
 end if
 thumbnail.bytes.Writefile(tmp_img)
 end if
End Sub

```

## 8.0 Components First Introduced in Firmware v3.0

The following section provides details for all the platform specific components introduced in firmware version 3.0.

### 8.1 roSocketAddress

The roSocketAddress is used by the roStreamSocket and roDataGramSocket components for TCP and UDP traffic respectively. roSocketAddress currently only support ipv4 addresses.

This object is created without any arguments:

- `CreateObject("roSocketAddress")`

`String getAddress()`

- Returns a string representing the ipv4 address in dotted decimal:port format.
- Example: "192.168.1.120:8888"

`Boolean setAddress(String address)`

- Sets the ipaddress to the dotted decimal:port string Example: "192.168.1.120:8888" or hostname Example: "roku.com".
- DNS lookup done if address is in hostname format.
- Returns true on success.

`Integer getPort()`

- Returns an Integer that is the 32 bit ipv4 port value.

`Boolean setPort(Integer port)`

- Sets the ipv4 port value to the passed 32 bit Integer value.
- Returns true on success.

`String getHostName()`

- Returns the string value of the hostname the component is set to.

`Boolean setHostName()`

- Sets the string value of the hostname of the component
- Returns true on success.

`Boolean isValidAddress()`

- Returns true if the component contains a valid ipAddress.
- Does a DNS lookup if the component currently has a hostname but not an ipAddress. And returns the result after the lookup.

### 8.2 roStreamSocket

The roStreamSocket component enables Brightscript apps to accept and connect to TCP streams as well as send and receive data with them. The interface is modeled on and works much like standard Berkeley sockets.

This object is created without any arguments:

- `CreateObject("roStreamSocket")`

#### Interface: ifSocketConnection

Each of these operations except listen() is either synchronous or asynchronous as determined by the socket's blocking behavior. If there is a valid assigned MessagePort, the blocking behavior is considered Asynchronous. Otherwise, the blocking behavior is considered Synchronous.

`Boolean listen(Integer backlog)`

- Returns true if listen can be done (generally if bound address is valid)

`Boolean isListening()`

- Returns true if listen has been successfully called on this socket

`Boolean connect()`



- Returns true if connect is successful.
  - Still might not be complete if socket is non-blocking
- `Object accept()`
- Returns an `roStreamSocket` if connection is pending, invalid otherwise.
  - Use status to distinguish among success (`eSuccess()` or `isConnected()`), not ready (`eOK()`), and error.
- `Boolean isConnected()`
- Returns true if a connect or accept has successfully completed on this socket

### Interface: `ifSocket`

These are the basic binding and data transfer operations used on both `roStreamSocket` and `roDataGramSocket`. They are synchronous or asynchronous as determined by the socket's blocking behavior. If there is a valid assigned `MessagePort`, the blocking behavior is considered Asynchronous. Otherwise, the blocking behavior is considered Synchronous.

- `Integer send(roByteArray data, Integer start, Integer length)`
- Returns the number of bytes actually written.
- `Integer send(String data)`
- Sends the whole string if possible.
  - Returns the number of bytes actually written.
- `Integer receive(roByteArray data, Integer start, Integer length)`
- Returns the number of bytes actually read.
- `Integer receive(String data, Integer length)`
- Tries to fill string of indicated length.
  - Returns length of String actually read from buffer.
- `Void close()`
- Performs an orderly close of socket.
  - After a close, most operations on the socket will return invalid.
  - On blocking sockets, this clears the receive buffer and blocks until the send buffer is emptied. Neither buffer may be read or written afterward.
  - On non-blocking sockets, both the send and the receive buffer may be read but not written.
- `Boolean setAddress(Object sockAddr)`
- `sockAddr` is an `roSocketAddress`.
  - Returns true if successfully set address using a BSD `bind()` call.
- `Object getAddress() as roSocketAddress`
- Returns `roSocketAddress` object bound to this socket.
- `Boolean setSendToAddress(Object sockAddr)`
- Set remote address for next message to be sent
  - `sockAddr` is an `roSocketAddress`
  - Returns true if successfully stored address as first half of underlying BSD `sendto()` call
- `Object getSendToAddress()`
- Returns `roSocketAddress` for remote address of next message to be sent
  - Can also be used to return the remote address on newly accepted sockets
- `Object getReceivedFromAddress() as roSocketAddress`
- Returns `roSocketAddress` for remote address of last message received via `receive()`
  - Can also be used to return the remote address on newly accepted sockets
- `Integer getCountRcvBuf()`
- Returns the number of bytes in the receive buffer
- `Integer getCountSendBuf()`
- Returns the number of bytes in the send buffer
- `Integer status()`
- Returns the `errno` of the last operation attempted or zero if the last operation was a

success.

### Interface: ifSocketAsync

The ifSocketAsync interface provides asynchronous socket features that utilize a full-featured select loop in the firmware that communicates to the application using a BrightScript MessagePort. This interface is valid on roStreamSocket and roDataGramSocket objects that were assigned a BrightScript port via SetMessagePort().

```
Boolean isReadable()
 ▪ Returns true if underlying select determines non-blocking read is possible
Boolean isWritable()
 ▪ Returns true if underlying select determines non-blocking write is possible
Boolean isExceptional()
 ▪ Returns true if underlying select determines non-blocking read of OOB data is possible
Void notifyReadable(Boolean enable)
 ▪ Enable roSocketEvent events to be sent via the message port when the underlying
 socket becomes readable.
Void notifyWritable(Boolean enable)
 ▪ Enable roSocketEvent events to be sent via the message port when the underlying
 socket becomes writable.
Void notifyError(Boolean enable)
 ▪ Enable roSocketEvent events to be sent via the message port when the underlying
 socket gets an exception or OOB data.
String getID()
 ▪ Returns a unique identifier that can be compared to the value returned by
 roSocketEvent.getSocketID() to match the underlying socket that the event is for.
```

### Interface: ifStreamSend

The ifStreamSend interface methods for sending data on a generic stream objects.

```
Void SendByte(Integer byte)
 ▪ Send the least significant byte of the passed in Integer. There is no unsigned char type in
 Bytescript so the actual byte that is sent is passed through the BrightScript interface as
 an Integer.
Void SendLine(String line)
 ▪ Send the passed in line string followed by an end of line marker on the socket.
 ▪ The end of line marker is '\r\n'
Void SendBlock(String block)
 ▪ Send the passed in block string on the socket.
```

### Interface: ifStreamRead

The ifStreamRead interface methods for sending data on a generic stream objects.

```
Integer Read_Byte()
 ▪ Read the next byte on the socket and return it as an Integer to BrightScript. There is no
 unsigned char type in Bytescript so the Integer returned is the result of promoting the
 byte read from the socket.
 ▪ Synchronous call blocks if no data available
 ▪ Returns -1 if socket closed.
Integer Read_Byte_If_Available(String line)
 ▪ Read the next byte on the socket and return it as an Integer to BrightScript. There is no
 unsigned char type in Bytescript so the Integer returned is the result of promoting the
 byte read from the socket.
 ▪ Asynchronous call returns -1 if no data available.
String Read_Line()
```

- Return the string that is the next data on the socket up to the end of line marker.
  - The end of line marker is '\r\n' by default
- String Read\_Block(Integer size)
- Read the next size bytes of data on the socket and return as a string.

#### Interface: ifSocketStatus

- Boolean eAgain()
- Return true if errno is EAGAIN.
- Boolean eAlready()
- Return true if errno is EALREADY.
- Boolean eBadAddr()
- Return true if errno is EBADADDR.
- Boolean eDestAddrReq()
- Return true if errno is EDESTADDRREQ.
- Boolean eHostUnreach()
- Return true if errno is EHOSTUNREACH.
- Boolean eInvalid()
- Return true if errno is EINVAL.
- Boolean eInProgress()
- Return true if errno is EINPROGRESS.
- Boolean eWouldBlock()
- Return true if errno is EWOULDBLOCK.
- Boolean eSuccess()
- Return true if errno is 0 (no errors).
- Boolean eOK()
- Return true if errno is has no hard error, but there could be async conditions: EAGAIN, EALREADY, EINPROGRESS, EWOULDBLOCK

#### Interface: ifSocketConnectionStatus

- Boolean eConnAborted()
- Return true if errno is ECONNABORTED.
- Boolean eConnRefused()
- Return true if errno is ECONNREFUSED.
- Boolean eConnReset()
- Return true if errno is ECONNRESET.
- Boolean eIsConn()
- Return true if errno is EISCONN.
- Boolean eNotConn()
- Return true if errno is ENOTCONN.

#### Interface: ifSocketOption

- Integer getTTL()
- Return the integer TTL (Time To Live) value for all IP packets on the socket.
- Boolean setTTL(Integer ttl)
- Set the integer TTL (Time To Live) value for all IP packets on the socket.
  - Returns true if successfully set.
- Boolean getReuseAddr()
- Return true if an address that has been previously assigned can be immediately reassigned.
- Boolean setReuseAddr(Boolean reuse)
- Set the whether an address that has been previously assigned can be immediately reassigned.
  - Returns true if successfully set.
- Boolean getOOBInline()
- Return true if Out Of Bounds data is read inline with regular data.

```

Boolean setOOBInline(Boolean inline)
 ▪ Set whether OOB data is received in regular read.
 ▪ Returns true if successfully set.
Integer getSendBuf()
 ▪ Return the current send buffer size.
Boolean setSendBuf(as Integer)
 ▪ Set the current send buffer size.
 ▪ Returns true if successfully set.
Integer getRcvBuf()
 ▪ Return the current receive buffer size.
Boolean setRcvBuf(as Integer)
 ▪ Set the current receive buffer size.
 ▪ Returns true if successfully set.

```

### Interface: **ifSocketConnectionOption**

```

Boolean getKeepAlive()
 ▪ Return true if keep alive is set.
Boolean setKeepAlive(Boolean enable)
 ▪ Enable keep alive if enable is true, otherwise disable it.
 ▪ If keep alive set, occasional no-data packets will be sent to keep the connection alive.
 ▪ Returns true if successfully set.
Integer getLinger()
 ▪ Return the max time in seconds that the socket close() blocks to allow send data to be
 flushed in synchronous mode.
Boolean setLinger(Integer time)
 ▪ Set the max time in seconds that the socket close() blocks to allow send data to be
 flushed in synchronous mode.
 ▪ Returns true if successfully set.
Integer getMaxSeg()
 ▪ Return the max TCP segment size.
Boolean setMaxSeg(Integer time)
 ▪ Set the max TCP segment size.
 ▪ Returns true if successfully set.
Boolean getNoDelay()
 ▪ Return true if no delay is on.
 ▪ With no delay on, data is sent as soon as it is available rather than waiting for enough
 data to fill a segment.
Boolean setNoDelay(Boolean enable)
 ▪ Enable the No Delay property on the socket.
 ▪ If No Delay is set, data is sent as soon as it is available rather than waiting for enough
 data to fill a segment.
 ▪ Returns true if successfully set.

```

### Events: **roSocketEvent**

The firmware's internal select loop fires this event where the `getSocketID` predicate indicates the async socket (`ifSocketAsync`) whose status has changed. The socket must enable specific event notifications via the `notify` methods of `ifSocketAsync`.

```

getSocketID()
 ○ Returns the socket id this Event is for.
 ○ Analogous to indicating a file descriptor in select() call with a status change.
 ○ Use ifSocketStatus or ifSocketConnectionStatus on the indicated socket to query
 the new status for the ifSocketAsync socket whose getID() value matches the
 event getSocketID() value.

```

### Example Code: roStreamSocket

```
' synchronous print to TCP server on port 54321
' uses Brightscript internal print via an interface

Function TCPPrint()
 tcp = createobject("roStreamSocket")
 if tcp<>invalid
 addr = createobject("roSocketAddress")
 addr.setAddress("10.1.1.2:54321")
 tcp.connect(addr) ' implied bind to arbitrary local port
 if tcp.isConnected()
 uniqueDev = createobject("roDeviceInfo").GetDeviceUniqueId()
 message = "Message from " + uniqueDev
 print #tcp message ' uses ifStreamSend
 else
 print "Couldn't connect"
 end if
 end if
 ' close will happen as tcp goes out of scope
End Function

' TCP asynchronous echo server on port 54321
' accepts multiple simultaneous client connections
' echoes whatever is received back to the corresponding client

Function InitConnection(tcp as Dynamic, msgPort as Object) as Dynamic
 ' tracks client connections
 if type(tcp)="roStreamSocket"
 this = createobject("AssociativeArray")
 this.tcp = tcp
 tcp.notifyReadable(true)
 this.buf = createobject("roByteArray")
 this.echoPos = 0
 else
 this = invalid
 end if
 return this
End Function

Function TCPEchoServer()
 msgPort = createobject("roMessagePort")
 connections = createobject("AssociativeArray")

 tcpListen = createobject("roStreamSocket")
 tcpListen.setMessagePort(mp) ' notifications for tcpListen goto mp
 addr = createobject("roSocketAddress")
 addr.setPort(54321)
 tcpListen.setAddress(addr) ' bind to an arbitrary host address
 tcpListen.notifyReadable(true)
 tcpListen.listen(4) ' no more than 4 pending connections

 continue = tcpListen.eOK() ' exit if listen gets error
 timeout = 1 * 10 * 1000 ' ten secs in milliseconds

 while continue
```

```

event = wait(msgPort,timeout)
if type(event)="roSocketEvent"
 changedID = event.getSocketID()
 if changedID=tcpListenID
 if tcpListen.isReadable()
 newConn = tcpListen.accept()
 if newConn<>invalid
 newID = Stri(newConn.getID()).trim()
 connections[newID] = InitConnection(newConn,msgPort)
 end if
 end if
 continue = tcpListen.eOK() ' exit if listen gets error
 else
 id = Stri(changedID).trim()
 conn = connections[id]
 if conn<>invalid
 close = false
 needSend = conn.buf.count() - conn.echoPos
 if needSend > 0
 if conn.tcp.isWritable()
 sent = conn.tcp.send(conn.buf, echoPos, needSend)
 if sent>0
 conn.echoPos = conn.echoPos + sent
 else if sent=0 ' client closed
 close = true
 end if
 if conn.echoPos>=conn.buf.count()
 conn.tcp.notifyWritable(false)
 conn.tcp.notifyReadable(true)
 end if
 end if
 else if conn.tcp.isReadable()
 received = conn.tcp.receive(conn.buf,0,512)
 if received>0
 input = conn.buf.ToAsciiString()
 print "Echo input: "; input; ""
 conn.tcp.notifyWritable(true)
 conn.tcp.notifyReadable(false)
 conn.echoPos = 0
 else if received=0 ' client closed
 close = true
 end if
 end if
 if close or not conn.tcp.eOK()
 conn.tcp.close()
 connections.delete(id)
 end if
 end if
 end for
else if message=invalid
 ' could weed out expired connections
 print "No activity"
end if
end while

tcpListen.close()
for each id in connections

```

```

 conn = connections[id]
 if conn<>invalid then conn.tcp.close()
 end for

End Function

' TCP client connecting to port 54321
' periodically sends message, waits for echo

Function TCPClient()
 msgPort = createobject("roMessagePort")
 tcp = createObject("roStreamSocket")
 tcp.setMessagePort(mp) ' all notifications come to mp
 addr = createobject("roSocketAddress")
 addr.setAddress("10.1.1.1:54321")
 tcp.setSendToAddress(addr) ' direct at server IP and port
 tcp.notifyWritable(true)
 tcp.connect()

 outBuf = createobject("roByteArray")
 outBuf.FromAsciiString("stream")
 totalSent = 0
 inBuf = createobject("roByteArray")
 continue = tcp.eOK()
 timeout = 1 * 10 * 1000 ' ten secs in milliseconds

 while continue
 event = wait(mp,timeout)
 if type(event)="roSocketEvent"
 changed = event.getID
 if changed=tcp.getID()
 continue = tcp.eOK()
 if tcp.isReadable()
 received = tcp.receive(inBuf,0,2048)
 if received>0
 input = inBuf.ToAsciiString()
 print "Echo input: "; input; ""
 tcp.notifyReadable(false)
 else if received=0 ' server closed
 continue = false
 end if
 else if tcp.isWritable()
 if tcp.isConnected()
 needSend = outBuf.count() - totalSent
 if needSend>0
 sent = tcp.send(outBuf,totalSent,needSend)
 if sent>0
 totalSent = totalSent + sent
 if totalSent>=outBuf.count()
 tcp.notifyReadable(true)
 tcp.notifyWritable(false)
 end if
 else if sent=0 ' server closed
 continue = false
 end if
 end if
 end if
 else

```

```

 tcp.connect()
 end if
end if
end if
else if message=invalid
 print "timeout"
 totalSent = 0
 tcp.notifyWritable(true) ' periodic send
end if
continue = continue and tcp.eOK()
end while

tcp.close()

```

### 8.3 roDataGramSocket

The roDataGramSocket component enables Brightscript apps to send and receive UDP packets. The interface is modeled on and works much like standard Berkeley sockets.

This object is created without any arguments:

- CreateObject("roDataGramSocket")

#### Interface: ifSocket

These are the basic binding and data transfer operations used on both roStreamSocket and roDataGramSocket. They are synchronous or asynchronous as determined by the socket's blocking behavior. If there is a valid assigned MessagePort, the blocking behavior is considered Asynchronous. Otherwise, the blocking behavior is considered Synchronous.

Integer send(roByteArray data, Integer start, Integer length)

- Returns the number of bytes actually written.

Integer send(String data)

- Sends the whole string if possible.
- Returns the number of bytes actually written.

Integer receive(roByteArray data, Integer start, Integer length)

- Returns the number of bytes actually read.

Integer receive(String data, Integer length)

- Tries to fill string of indicated length.
- Returns length of String actually read from buffer.

Void close()

- Performs an orderly close of socket.
- After a close, most operations on the socket will return invalid.
- On blocking sockets, this clears the receive buffer and blocks until the send buffer is emptied. Neither buffer may be read or written afterward.
- On non-blocking sockets, both the send and the receive buffer may be read but not written.

Boolean setAddress(Object sockAddr)

- sockAddr is an roSocketAddress.
- Returns true if successfully set address using a BSD bind() call.

Object getAddress() as roSocketAddress

- Returns roSocketAddress object bound to this socket.

Boolean setSendToAddress(Object sockAddr)

- Set remote address for next message to be sent
- sockAddr is an roSocketAddress
- Returns true if successfully stored address as first half of underlying BSD sendto() call



```
Object getSendToAddress()
 ▪ Returns roSocketAddress for remote address of next message to be sent
 ▪ Can also be used to return the remote address on newly accepted sockets
Object getReceivedFromAddress() as roSocketAddress
 ▪ Returns roSocketAddress for remote address of last message received via receive()
 ▪ Can also be used to return the remote address on newly accepted sockets
Integer getCountRcvBuf()
 ▪ Returns the number of bytes in the receive buffer
Integer getCountSendBuf()
 ▪ Returns the number of bytes in the send buffer
Integer status()
 ▪ Returns the errno of the last operation attempted or zero if the last operation was a success.
```

### Interface: ifSocketAsync

The ifSocketAsync interface provides asynchronous socket features that utilize a full-featured select loop in the firmware that communicates to the application using a BrightScript MessagePort. This interface is valid on roStreamSocket and roDataGramSocket objects that were assigned a BrightScript port via SetMessagePort().

```
Boolean isReadable()
 ▪ Returns true if underlying select determines non-blocking read is possible
Boolean isWritable()
 ▪ Returns true if underlying select determines non-blocking write is possible
Boolean isExceptional()
 ▪ Returns true if underlying select determines non-blocking read of OOB data is possible
Void notifyReadable(Boolean enable)
 ▪ Enable roSocketEvent events to be sent via the message port when the underlying socket becomes readable.
Void notifyWritable(Boolean enable)
 ▪ Enable roSocketEvent events to be sent via the message port when the underlying socket becomes writable.
Void notifyError(Boolean enable)
 ▪ Enable roSocketEvent events to be sent via the message port when the underlying socket gets an exception or OOB data.
String getID()
 ▪ Returns a unique identifier that can be compared to the value returned by roSocketEvent.getSocketID() to match the underlying socket that the event is for.
```

### Interface: ifSocketStatus

```
Boolean eAgain()
 ▪ Return true if errno is EAGAIN.
Boolean eAlready()
 ▪ Return true if errno is EALREADY.
Boolean eBadAddr()
 ▪ Return true if errno is EBADADDR.
Boolean eDestAddrReq()
 ▪ Return true if errno is EDESTADDRREQ.
Boolean eHostUnreach()
 ▪ Return true if errno is EHOSTUNREACH.
Boolean eInvalid()
 ▪ Return true if errno is EINVAL.
Boolean eInProgress()
 ▪ Return true if errno is EINPROGRESS.
Boolean eWouldBlock()
 ▪ Return true if errno is EWOULDBLOCK.
```

```
Boolean eSuccess()
 ▪ Return true if errno is 0 (no errors).
Boolean eOK()
 ▪ Return true if errno is has no hard error, but there could be async conditions: EAGAIN,
 EALREADY, EINPROGRESS, EWOULDBLOCK
```

### Interface: ifSocketOption

```
Integer getTTL()
 ▪ Return the integer TTL (Time To Live) value for all IP packets on the socket.
Boolean setTTL(Integer ttl)
 ▪ Set the integer TTL (Time To Live) value for all IP packets on the socket.
 ▪ Returns true if successfully set.
Boolean getReuseAddr()
 ▪ Return true if an address that has been previously assigned can be immediately
 reassigned.
Boolean setReuseAddr(Boolean reuse)
 ▪ Set the whether an address that has been previously assigned can be immediately
 reassigned.
 ▪ Returns true if successfully set.
Boolean getOOBInline()
 ▪ Return true if Out Of Bounds data is read inline with regular data.
Boolean setOOBInline(Boolean inline)
 ▪ Set whether OOB data is received in regular read.
 ▪ Returns true if successfully set.
Integer getSendBuf()
 ▪ Return the current send buffer size.
Boolean setSendBuf(as Integer)
 ▪ Set the current send buffer size.
 ▪ Returns true if successfully set.
Integer getRcvBuf()
 ▪ Return the current receive buffer size.
Boolean setRcvBuf(as Integer)
 ▪ Set the current receive buffer size.
 ▪ Returns true if successfully set.
```

### Interface: ifSocketCast

Only the roDataGramSocket component supports the ifSocketCast multicast interface. The roStreamSocket component does not support multicast.

```
Boolean getBroadcast()
 ▪ Return true if broadcast messages are enabled to be sent or received.
Boolean setBroadcast(Boolean enable)
 ▪ If enable is true, enable broadcast messages to be sent or received; otherwise do not
 send or receive broadcast messages.
 ▪ Returns true if successfully set.
Boolean joinGroup(Object ipAddress),
 ▪ Join the multicast group specified by the passed in multicast ipAddress.
 ▪ Ipv4 multicast addresses are in the range: 224.0.0.0 through 239.255.255.255
 ▪ Returns true if successfully set.
Boolean dropGroup(as roSocketAddress)
 ▪ Drop out of the multicast group specified by the passed in multicast ipAddress.
 ▪ Ipv4 multicast addresses are in the range: 224.0.0.0 through 239.255.255.255
 ▪ Returns true if successfully set.
Boolean getMulticastLoop()
 ▪ Return true if multicast messages are enabled for local loopback.
 ▪ If enabled, multicast message sent locally are to be received locally.
```

```
Boolean setMulticastLoop(Boolean enable)
```

- If enable is true, enable local loopback of multicast messages; otherwise do not send or receive broadcast messages.
- Returns true if successfully set.

```
Integer getMulticastTTL()
```

- Return the TTL integer value for multicast messages.
- TTL is the number of hops a packet is allowed before a router drops the packet.

```
Boolean setMulticastTTL(Integer ttl)
```

- Set the TTL integer value for multicast messages.
- TTL is the number of hops a packet is allowed before a router drops the packet.
- Returns true if successfully set.

### Events: roSocketEvent

The firmware's internal select loop fires this event where the getSocketID predicate indicates the async socket (ifSocketAsync) whose status has changed. The socket must enable specific event notifications via the notify methods of ifSocketAsync.

```
getSocketID()
```

- Returns the socket id this Event is for.
- Analogous to indicating a file descriptor in select() call with a status change.
- Use ifSocketStatus or ifSocketConnectionStatus on the indicated socket to query the new status for the ifSocketAsync socket whose getID() value matches the event getSocketID() value.

### Example Code: roDataGramSocket

```
' UDP 2-way peer-to-peer asynchronous comm on port 54321
' periodically sends out a message to a specific address and port
' prints any message it receives

Function UDPPeer()
 msgPort = createobject("roMessagePort")

 udp = createobject("roDatagramSocket")
 udp.setMessagePort(msgPort) 'notifications for udp come to msgPort
 addr = createobject("roSocketAddress")
 addr.setPort(54321)
 udp.setAddress(addr) ' bind to all host addresses on port 54321
 addr.SetHostName("10.1.1.1")
 udp.setSendToAddress(addr) ' peer IP and port
 udp.notifyReadable(true)

 timeout = 1 * 10 * 1000 ' ten seconds in milliseconds
 uniqueDev = createobject("roDeviceInfo").GetDeviceUniqueId()
 message = "Datagram from " + uniqueDev
 udp.sendStr(message)
 continue = udp.eOK()

 while continue
 event = wait(msgPort,timeout)
 if type(event)="roSocketEvent"
 if event.getSocketID()=udp.getID()
 if udp.isReadable()
 message = udp.receiveStr(512) ' max 512 characters
 print "Received message: "; message; ""
 end if
 end if
 end if
 end while
end Function
```

```
 end if
 else if event=invalid
 print "Timeout"
 udp.sendStr(message) ' periodic send
 end if
end while

udp.close() ' would happen automatically as udp goes out of scope
End Function
```