Smart TV Reference Application
Developer Guidelines

Prepared By:

Brightcove, Inc.

Revision 1.0

Contents

Overview

This is an LG video application that displays several playlists from a Brightcove account. The application has been designed so minimal configuration is necessary, and allows customers to make customizations to change the UI or add/remove features according to the functionality they desire..

The general feature set includes:

- Browsing multiple playlists
- Playback of videos from the playlists
- CSS for layout and theming
- Google Analytics integration
- Ability to view related videos when current video is playing
- Advertising integration with YuMe

Prerequisites

The application supports either Progressive Download (PD) for HTTP Live Streaming (HLS) for video playback. If PD is configured then the application will look for renditions, which are at least 1024 in width. The media delivery cannot be mixed it is assumed the account will be setup properly for whatever method is configured.

Configuration

The application has several configuration options, which are located in **config.js**. The following table outlines the options and their purpose.

Config Option	Value	Description
lang	language code	The language to be used for localization of text within the application. The application has been localized with the following codes: en, en-gb, en-ca, es, fr and de. See the section on Localization for more information.
app_name	name of application	The name of the application. Currently used only in the reporting of analytic

		events.
ga_token	Google Analytics token	The Google Analytics token for the account which reporting calls should be made to.
media_delivery	“http” or “http_ios”	This controls whether the media delivery is going to be PD or HLS. *Note: Please make sure the Brightcove account is properly set up for the media delivery type that’s specified here.*
playlist_ids	array of playlist ids	The list of playlists that the application is going to display. Recommended number of videos in each playlist does not exceed 30 videos.
continuous_play	“playlist”, “all”, or false	Enables continuous play functionality. If the value is set to “playlist”, all videos within a playlist will play, beginning at the video from which the player was launched, before returning to the main screen. If the value is set to “all”, all videos within a playlist will play, beginning at the video from which the player was launched, and automatically continue onto the next playlist once all videos in the previous playlist have completed. If the value is set to false (boolean), the user will be returned to the menu after each video completes. Both the “all” and “playlist” values also apply to playlists within the info window. Once all videos within an info window playlist have played, the user will be returned to the main screen even if the “all” setting is used. “all” will not trigger another playlist to play in this scenario.
token	Brightcove API token	The Brightcove media API token to be used by the application. This needs to be a read token with URL access.
ad_server_url	URL	A URL that will supply advertising data.

		The response must be JSON.
preroll_ads	true or false	Turns the pre-roll functionality on or off.
yume_domain_id	domain ID	Your YuMe domain ID to YuMe knows which advertisement to play.
yume_qs_params	Query string parameters	Additional query string parameters (if necessary) to send along to YuMe during an advertisement request.
yume_preroll_playlist	Dynamic playlist JSON file	The playlist URL to hit (via AJAX) on YuMe's server.
title_ad_play_interval	pre-roll interval number	The interval (in videos) at which a pre-roll should play.

Localization

The video metadata that is displayed in the application is hosted by Brightcove and is delivered via the Media API. Any localization that needs to happen on this data should be controlled at the account level and is beyond the scope of this document. Typically separate accounts are created to host videos with different localized metadata.

The application has several labels and messages, which are currently localized for English, Spanish, French and German. The **localized-copy.js** file can be easily extended to support other languages by adding their respective language codes.

Note: *In the current application the* `shortDescription` *video field is automatically truncated to 175 characters in order to fit within various parts of the application. To avoid truncation the metadata Brightcove account should be set-up so that it is no longer than 175 characters.*

Application Structure

HTML

All HTML is located inside the index.html file. HTML for the playlists (List class), playlist items (Row class), and info window popup (Info) class are created dynamically using the Handlebars template engine.

CSS

All CSS is located inside the style.css file, and is organized into 8 sections:

- Main elements (body, overlay, player, logo, etc.)
- Playlist navigation
- Error message window
- Info window
- Main/browse screen playlists
- Playlist items
- Spinner
- Video player controls
- Buttons & navigation links

Aside from general color customizations, two images should be changed (both are located towards the top of the file):

1. The background image (located in `#content`). Either change the background declaration to a new image, or replace the `background.jpg` file.
2. The logo on the top-left corner of the application (located in the `#logo` CSS declaration).

JavaScript

The JavaScript is organized into 10 separate files:

Application.js

This is the main, top-level file responsible for controlling the rest of the application. Its responsibilities include:

- Initializing the video player, list, and info classes.
- Pre-loading hover state images for the magic remote.
- Retrieving all playlists from Brightcove and building the UI.
- Listening for all mouse/keydown events and invoking the appropriate action. Often times this means calling a method on a class, or on an instance of a class.
- Listening for all events triggered by the video player class and invoking the appropriate action.

The entire application is initialized inside the `init()` function on document ready.

List.js

List.js contains all the logic necessary for navigating a playlist on both the main and browse screens. When the up/down arrows are clicked or pressed, List.js updates the active playlist and creates a new instance of the “row” class for that playlist, if one has not already been created.

Row.js

Row.js is responsible for navigating videos within a playlist. An instance of this class is initialized for each playlist & related videos, but only when the playlist becomes active. The HTML for that row is built and inserted into the DOM when a new row is initialized. The application.js file detects when the right/left arrows are pressed or clicked, and hands it off to the info/list instance for that playlist. If the info window is open, the info instance receives the call, and if the main screen or browse screen is visible, the list instance receives the call. In either case, the call is again handed off to the active row instance within the playlist to move the slide forward or backwards.

Info.js

Much like List.js and Row.js, Info.js manages the display & functionality of the info window popup. When a new List instance is created for a playlist, a new Info instance is also created, which fetches data & inserts a hidden container div into the DOM. DOM elements associated with the Info window are not re-used; rather, once an info window has been fetched & built, it persists for quicker re-opening later. Application.js controls which info window to open based on the active playlist.

VideoPlayer.js

This class is responsible for creating the HTML `<object>` tag that holds the video, retrieving the video from Brightcove via an AJAX call, and controlling all interaction with it (play, pause, etc.). A new video instance is created once inside the Application.js file.

If the application is run in an environment that supports the HTML5 video, this file also overwrites class methods to leverage the `<video>` tag and its API’s rather than the `<object>` tag.

When the application is run in a webkit-based browser on the desktop, the HTML5 `<video>` tag is used. When the application is run inside the LG box, the `<object>` tag is used. Do not assume any changes will work on the LG box if it works in the browser.

Config.js

Application-specific configuration options.

Localized-copy.js

Localization translations for all non-data-driven text within the application.

Events.js

This file contains an “events” object that can be extended into any other object. Row.js, List.js, and Info.js all extend this object to enable the publisher/subscriber design pattern. Each of these objects dispatch events at crucial points for other objects to respond to. For example, when a row instance changes its slide, it also dispatches an `indexChanged` event, passing along the index of the new slide to all its listeners. Instance of the list class listen for this event (via `row.addEventListener()`) to update metadata associated with the new slide.

Analytics.js

Logic associated with Google Analytics. This class is initialized with a Videoplayer object inside Application.js. Please see the Analytics section for a table of events that are reported.

Ad-policy.js

Logic responsible for retrieving advertising data from the ad server, and deteremining whether or not a pre-roll should play.

Yume-sdk.js

Yume’s advertising integration SDK.

User Interface

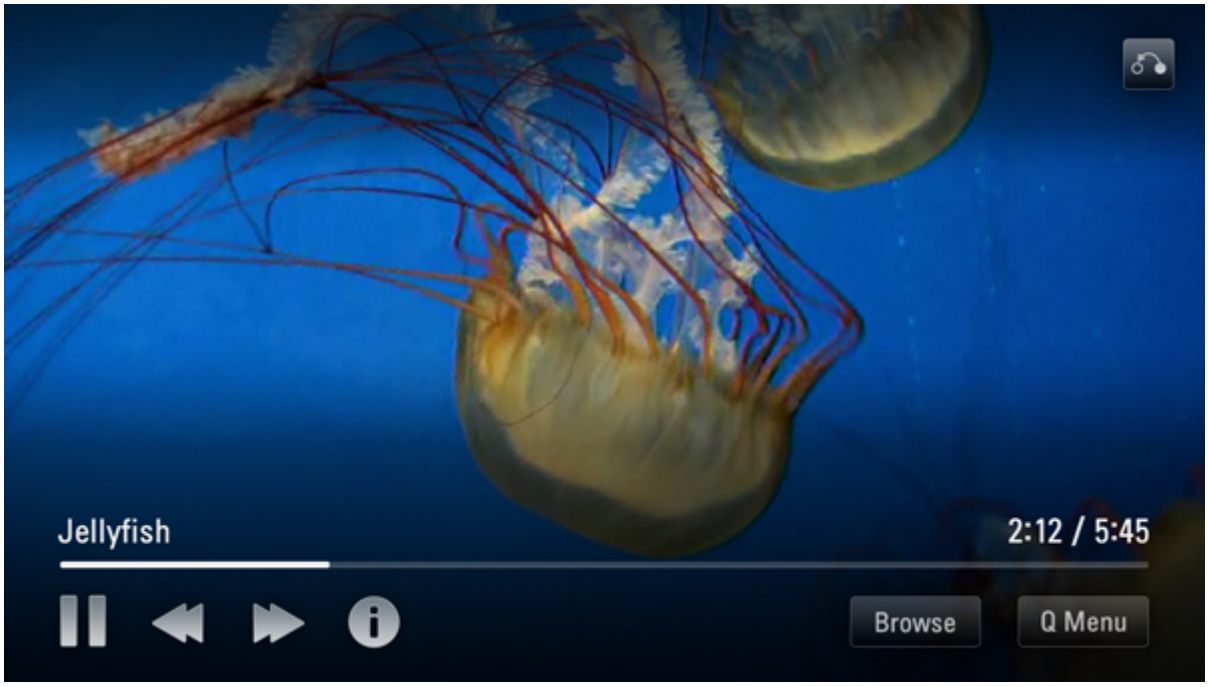
The UI of the application is composed of four sections:

Main Screen

This is the first screen that is displayed when the application starts up. Users can see a background image for the application as well as one open playlist.

Video Player

The video player screen appears when a video has been selected for playback. On screen controls display the status of the video such as the current time, time remaining and whether the video has been paused or not. The onscreen controls only appear when an action on a remote has been taken and then disappear after a time delay. Controls can be toggled on/off by pressing the “Enter” button on the remote.

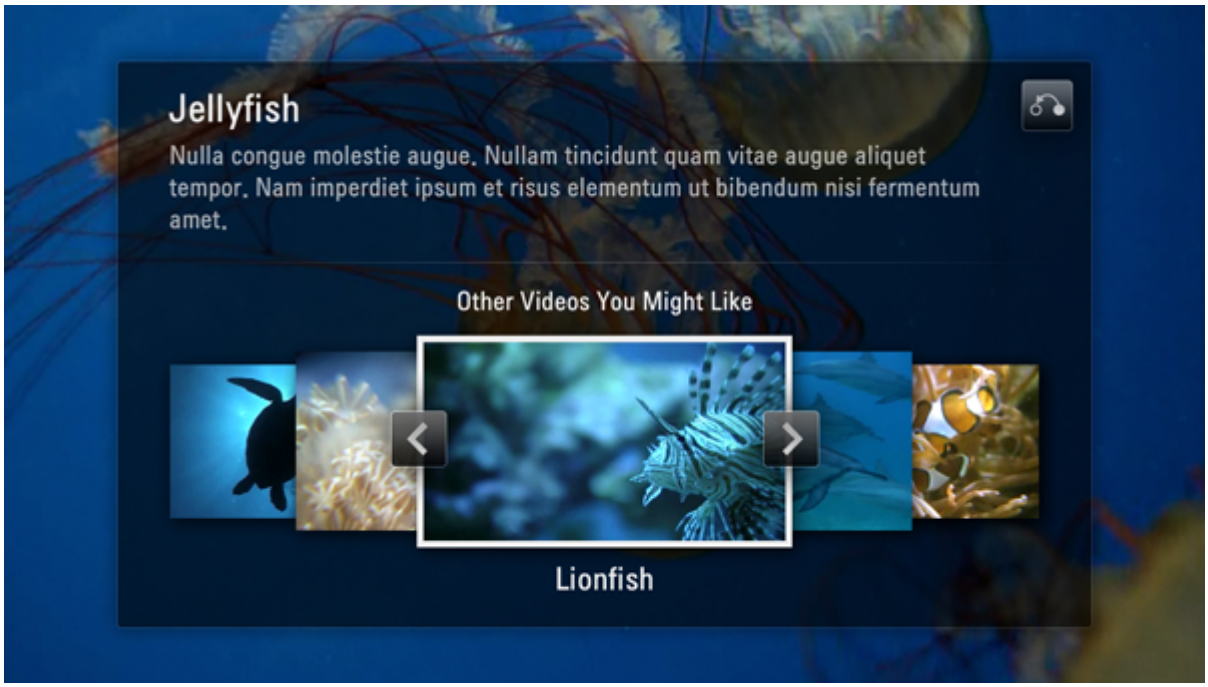


Browse Screen

The same playlist navigation that is displayed on the main screen cn also can be brought up by clicking the **BROWSE** button from the control button bar. The same behavior for the playlists and videos outlined before applies here. The video will remain playing in the background while the browse screen is open. Users can return to the video by pressing the remote's back button, by clicking the back button within the application, or by clicking the miniature controls box..

Info Screen

The info screen displays the title and short description for the currently playing video as well as related videos. The screen can be launched by clicking the **INFO** button on either the application or the remote.



Analytics

To handle analytics and reporting on video usage, the LG reference application includes an integration with Google Analytics (GA). All that is required for proper configuration is a valid GA token which usually comes in the form UA-XXXXX-X.

The following page views and events are tracked by default. Any additional tracking or changes would require customization of the analytics.js file.

Event Name	Data	Description
Video Load	Video ID	Event dispatched when a new video is loaded.
Media Begin	Video ID	Event dispatched when a new video starts playing.
Media Resume	Video ID	Event dispatched every time a video has started playing.
Media Pause	Video ID	Event dispatched every time a video has been paused.
Media Complete	Video ID, Time Watched	Event dispatched when a video has completed.
Ad Start	None	Event dispatched when an advertisement starts playing.
Ad Complete	None	Event dispatched when an advertisement finishes playing.
25% Milestone Passed	Video ID	Event dispatched when a video progresses across the 25% mark.
50% Milestone Passed	Video ID	Event dispatched when a video progresses across the 50% mark.
75% Milestone Passed	Video ID	Event dispatched when a video progresses across the 75% mark.

Advertising

The application supports pre-roll advertising out of the box with YuMe. Pre-rolls can be set to play every X number of videos. To enable this functionality, the following settings must be configured in the config.js file:

Setting	Description
preroll_ads	Boolean: turns the pre-roll functionality on or off
yume_domain_id	String: your YuMe domain ID
title_ad_play_interval	Number: the interval (in videos) at which a preroll should play

Other advertising-related settings may be changed to support your unique case, but the above settings are the minimum to be changed.

All functionality related to the fetching of data is in the `ad-policy.js` file. When a request to play a video is started, the video player class consults the ad policy on whether or not a pre-roll should play. Communication between the two classes is done with a callback. If a pre-roll should be played, the ad policy invokes the callback with the JSON returned from the `ad_server_url` AJAX call. Otherwise, the callback is invoked without any parameters.

How To Guide

This section describes how to create a production-ready version of the reference application.

Customizing the Application

1. Ensure at least two playlists have been created in your Brightcove Video Cloud account, and at least one video exists in each playlist. If PD is configured as the `media_delivery` type (see below) then the application looks for renditions, which are at least 1024 pixels in width.

   Note: *the application will operate as expected with one playlist, but the "next" and "previous" arrows will continue to appear, which likely is not desirable.*

2. Open the config.js file and at a minimum, set the following options:

Option	Description
`app_name`	The name of the application. This is reported to Google Analytics.
`ga_token`	Google analytics token
`media_delivery`	Either “http” for PD or “http_ios” for HLS.
`token`	Brightcove media API token. Must be read with URL access.
`playlist_ids`	Two or more playlists to display.

All other options are optional and are not required for the application to function properly.

3. Replace the `images/logo.png` file with your own logo. For best results the file should be 24bit with alpha transparency.

4. Replace the `images/background.jpg` file with your own background image. This file should be 1280x720 pixels in size.

5. Refer to the style.css file and the images directory for any other customizations.

Recommendations

- If you do not have an LG box to test on, we recommend that you only change the logo and background image. Any other changes (including colors) cannot be guaranteed to render the same as you see it in the browser.
- Do not attempt to change the font. The LG box will always overwrite your font-family declarations with its own font.

LG Requirements

Any changes made to the application should respect LG’s own requirements. Failure to do so could likely mean rejection from LG’s QA team after the application has been submitted:

- All buttons need to be accessible by the magic remote’s cursor mode and 4-way control pad.
- The exit button on the main page must call `window.NetCastBack()`.
- Once video playback begins and the controls disappear, the magic remote cursor must also disappear (via `window.NetCastMouseOff(0)`).

- The “OK” or “Enter” button on the remote should toggle the controls on/off during video playback.

Testing the Application

In a Webkit-based Browser

The application has been designed to be operational inside webkit-based browsers (Google Chrome or Safari) that support the HTML5 video tag. Therefore, it is possible to use the browser as a development and preliminary test environment. That said, there is some functionality that leverages the LG box API and therefore requires the LG box:

1. The “exit” button from the main screen will not work in the browser.
2. The mouse cursor will not disappear once the video controls disappear after playing a video in the browser.
3. The “qMenu” button will not work in the browser.

Furthermore, the browser and LG box render HTML elements differently, especially as it pertains to positioning and alignment. Do not assume a pixel-perfect layout in the browser will be pixel-perfect on the box.

On the LG Box

To test your application on LG test “CP” hardware:

a. Upload the application to a web server.
b. Ensure the LG box is properly connected to the Internet. To access the device’s network settings:
    i. Click the “Menu” key on the remote.
    ii. Navigate to & press the “Settings” button.
    iii. Press the world icon (second from the bottom)
c. On a USB thumb drive, create a folder called “test_cp”. Within this folder, create a file named “test_config.xml” with the following contents:

```
<?xml version='1.0'  encoding='UTF-8'?>
<test_cpinfo>
  <item id="Test" type="browser" testIcon="false">
    <id>1</id>
    <url>http://www.web-server.com/location-to-app/</url>
    <spatial_navi>0</spatial_navi>
    <virtual_keyboard>0</virtual_keyboard>
    <loading_msg>1</loading_msg>
    <keycode_type>1</keycode_type>
```

```
    <accept_lang>1</accept_lang>
    <progress>50</progress>
    <drm_cookie>0</drm_cookie>
    <default_lang>en</default_lang>
    <bg_color>1</bg_color>
    <mouseoff_event>0</mouseoff_event>
    <magic_rcu>1</magic_rcu>
    <exit_key>0</exit_key>
    <user_agent>0</user_agent>
    <resolution>0</resolution>
    <widget>0</widget>
  </item>
</test_cpinfo>
```

d. Change the value of the `<url>` node to the URL of your application. This is the only change you need to make.
e. Plug the USB drive into the LG box (USB port 1 - it will not work in ISB port 2).
f. If prompted to “Use the USB drive”, choose “No”.
g. Press the “Menu” button on the remote.
h. Under the “Premium” widget box, press the “+” icon button.
i. On the next screen, your application should be the only application displayed on the page, and it’ll have the title “test”.
    i. If you see your application, press enter to launch it.
    ii. If you don’t see your application, press the “Back” button on the remote, wait a few seconds, and try again. If this still doesn’t work:
        1. Ensure the USB drive is plugged into USB port 1.
        2. Ensure the box is connected to the correct wireless network.
        3. Power off/on the LG box and try again. It can sometimes take a few attempts before the box will recognize your application.

Engaging with LG

Once the application has been customized to your liking, it must be submitted to LG for testing and approval. Testing can take from 1-4 weeks. The registration process is located here: http://seller.lgappstv.com/seller/guide/sellerRegistration.lge