Web App Development Quick Start Guide (WebAppSample_MediaPluginVideoPlayer)

Version 1.0.0 – January 2013

LGDEV-087

Home Entertainment Company
LG Electronics, Inc.

Copyright

About This Document

Revision History

Document Version	Date	Comment
1.0.0	2013, January 10	Initial Release

Purpose

This document is a quick guide intended to describe how developers develop and test web applications using the Media Plugin Video Player Sample example.

Reference Documents

Refer to the following documents:
- Developer Library in LG Developer (http://developer.lge.com) website

Conventions

Codes

Source code and examples are indicated in the `grey Courier New` font.

Note, Caution

Note and caution are used to emphasize information.
The following samples describe when each is used.

NOTE

Contains information about something that is helpful to you.

CAUTION

Contains important information about something that you should know.

Abbreviation

The following table defines the abbreviations used in this document.

Abbreviation	Description
CSS	Cascading Style Sheet
SDK	Software Development Kit

Contents

Figures

Tables

1 Introduction

This chapter describes Media Plugin Video Player Sample and its structure.

1.1 Overview of Media Plugin Video Player Sample

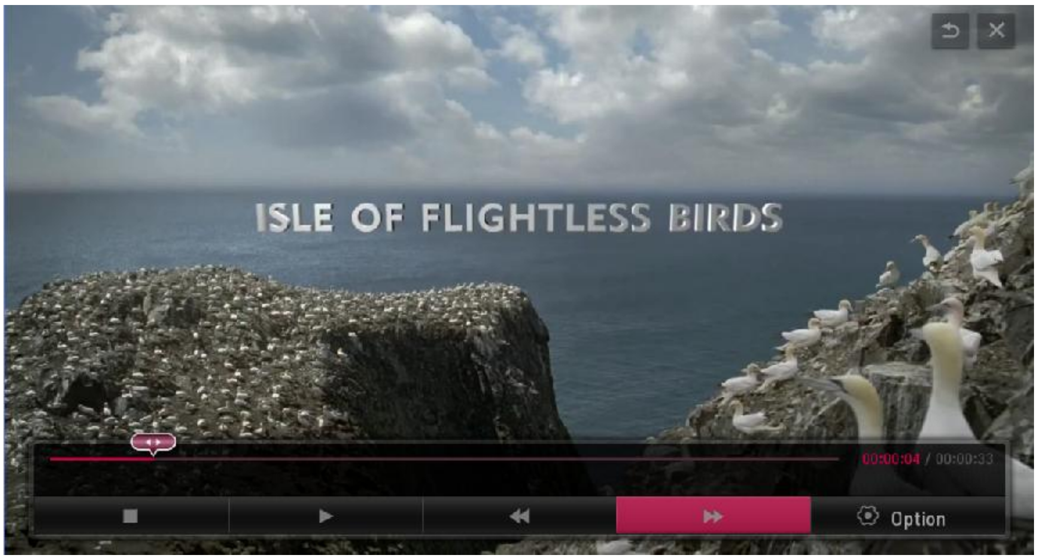
A web application that is developed based on general HTML standards has the following structure for its operation. If you separate each component independently, it is easy to manage application sources. Developers need to implement each component as necessary.

- **HTML: Defines application structure**
- **JavaScript: Defines application operation**
- **CSS: Defines application design elements**

In this guide, we are going to learn how to develop and test a media plugin video player web application. Media Plugin Video Player Sample uses the media object for displaying the video. This sample includes the basic functionalities of the media plugin video player like play, pause, forward and backward. The seek bar or the status bar comes by default.

This sample also provides RCU (Remote Control Unit) key navigation for control elements inside the sample. It will always focus the currently selected control/button.

The operation of media plugin video player sample application is shown below.

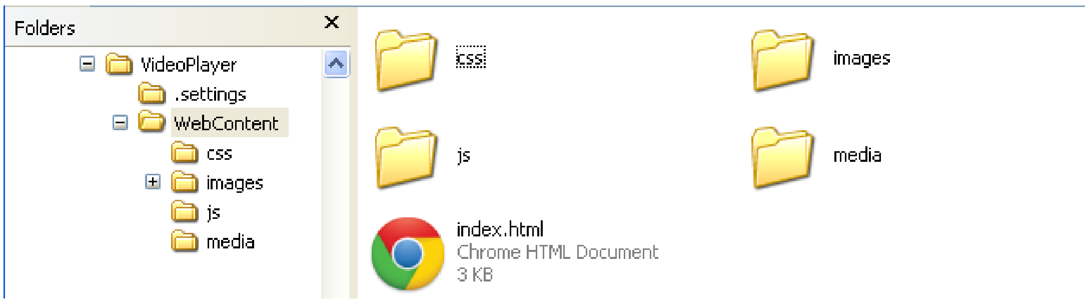


[Figure 1] Media Plugin Video Player Sample Application

1.2 Structure and Resources of Media Plugin Video Player Sample Project

The file structure of Media Plugin Video Player Application that is provided as a sample is shown below. You can download the sample codes from **[Resource Center > Smart TV > SDK & Tools > Tools & Samples]** menu in LG Developer (**http://developer.lge.com**) website.

You can either copy this sample source into the WebContent folder of the project that is created in LG IDE or enter the code directly by referring to the description in this guide.



[Figure 2] Structure and Resources of Media Plugin Video Player Sample Project

[Table 1] Structure of Media Plugin Video Player Sample Project

Folder			File	Description
Media Video Sample	Plugin Player		index.html	Initial operation file of an application
		css	player.css	Design elements of an application
		js	videoplayer.js	Media Plugin Video player playback control functionalities
			keycode.js	Key code definitions and value assignments

2 Setting Up the Project

Although it is okay to use a general editor to develop a web application, you can easily develop, debug, and test the application if you use the LG IDE. This chapter describes how to set up a project using the LG IDE.

Note

- For detailed information on how to start using LG IDE, refer to **Developing > Using SDK** section in Developer Library in LG Developer (**http://developer.lge.com**) website.

1. Open LG IDE and click **[File > New > LG Web Project].**

2. In the 'New LG Web Project' window, give a project name and select 'SDK Version for Web Application'. Then, click **[Finish].**

3. Add JavaScript, CSS, and image files.

Create each folder for JavaScript, CSS, or image files that will be used in this example in the WebContent folder. Although folders are not mandatory, it will be helpful to manage resources using folders when considering resources increase in the future.

Select and right-click the WebContent node and select **[File]** and then enter a file name including its extension in the file name field to add JavaScript or a CSS file.

if you want to use any external library for your application, add library file(in this case jquery-ui.js).

4. Now, a basic foundation is prepared to implement a sample example.

3 Implementing the Code

This chapter describes the description and expected results of Media Plugin Video Player Sample example.

Note

Refer to **Developing > API** section in Developer Library in LG Developer (**http://developer.lge.com**) website for detailed information on how to use Web APIs.

3.1 Descriptions on Source Codes

This section describes the core part of the sample source codes. The entire source codes are provided in the attached source zip file.

Linking JavaScript and CSS (index.html)

Specifies the path of JavaScript, style sheet and library(if any) source file that will be used in an application.

```
<link rel="stylesheet" href="css/player.css"/>
<script src="js/jquery-1.8.2.min.js"></script>
<script src="js/keycode.js"></script>
<script src="js/videoplayer.js"></script>
<script src="js/jquery-ui.js"></script>
```

Designing displaying structure (index.html)

Designs the structure to be displayed on an application screen using the <div> tag.

```
<div class="playerLayout">
Video player layout area

<div id="videoHolder">
    <object type="application/x-netcast-av"  width="1280" height="720"
          id="video">
    </object>
</div>
Video content area and declairation of video media plugin object with
width and height

<div class="playerBottom">
Bottom area of video player

<div class="playerButtonLayout">
Video player button layout area

<div class="progressBarLayout">
Progress bar and its components
</div>

<div id="buttonLayout">
Playback control buttons
</div>


<div id="ballCoverage">
      <div id="progressBall" class="progressBallInitial" > </div>
 </div>
Progress Ball Area

<div class="keyHelp" >
    <div class="backKey"></div> Back Button
    <div class="exitKey"></div> Exit Button
</div>
Area for Help Keys
```

Displaying the progress bar (index.html)

The status of the progress bar and the remaining time are updated as the video is being played.

Progress Bar Area is divided in five parts. First part is background, second part will show buffered position, third will show current status, fourth part is clicked area and last part will show remain time and total time.

```
<div class="progressBar">
          <div id="progressBg" class="progress progressBg"></div>
          <div id="progressBuffer" class="progress progressBuffer" ></div>
          <div id="progressBarStatus" class="progress
progressBarStatus"></div>
          <div id="progressBarClick" class="progress
progressBarClick"></div>
          <div class="runningTime"> <span id="remainingTime"></span> <span
id="totalTime" ></span> </div>
        </div>
```

Displaying the running movie information (index.html)

Displays the name and type of the running video.

```
<div class="runningMovieInfo">
          <div class="runningMovieName"> </div>
          <div class="runningMovieType"></div>
</div>
```

Displaying the button layout (index.html)

Contains various buttons like stop, play, rewind, forward and option.

```
<div id="buttonLayout">
        <ul>
          <li id="stop" class="stopButton" ><img
src="images/player_btn_icon/movie_btn_icon_stop_n.png" alt="stopBtn"
class="center"/></li> Stop button

         <li id="play" class="playButton"><img
src="images/player_btn_icon/movie_btn_icon_play_n.png" alt="playBtn"
class="center" /></li> Play button

         <li id="rewind" class="rewindButton"><img
src="images/player_btn_icon/movie_btn_icon_rewind_n.png" alt="rewindBtn"
class="center" /></li> Rewind button

         <li id="forward" class="forwardButton"><img
src="images/player_btn_icon/movie_btn_icon_forward_n.png" alt="forwardBtn"
class="center"/></li> Forward button

         <li id="option" class="optionButton">
            <div class="imgTextCenter"> <img
src="images/player_btn_icon/movie_btn_icon_option_n.png" alt="optionBtn"/>
<b class="textCenter">Option</b> </div> Option button

          </li>
        </ul>
      </div>
```

Implementing function for document ready (videoplayer.js)

Implements a JavaScript code which will execute on document ready event.

1. Getting the media plug-in object of video.
2. playMedia method plays & pauses the video depending on the current status of video.
3. Adding onPlayStateChange event listener for the play state change of video event and calling playStateChange method.
4. Adding onBuffering event listener and calling buffering method.
5. Calculating width of progress bar.

```
$('document').ready(function() {
  1  video = getVideo();
  2  playMedia();
  3  video.onPlayStateChange = playStateChange;
  4  video.onBuffering = buffering;
  5  progressBarWidth=$('#progressBg').width();

}
```

Binding key event to controls (videoplayer.js)

Implements a JavaScript code which will execute on key down event.

NetCastBack method is used to return to my app screen on click of return button.

setFocus method is used to change the color of the button from normal to focus based on index parameter (mouse over or remote keynavigation event).

selectedButton method is used to handle mouse click event or click of Enter button.

showPlayer method is used to show the player controls.

triggerHide method is used to call the hide function in 5 seconds.

```
$(document).keydown(function(event) {
       var key = event.keycode || event.which;
       switch(key){
          case VK BACK :
                       if(window.NetCastBack){
                         window.NetCastBack();
                       }
                       break;
          case VK RIGHT :
                       setFocus(1)
                       break;
          case VK LEFT :
                       setFocus(-1)
                       break;
          case VK UP :
                       if(rowID<4){
                         rowID++;
                       }
                       if(     video.playState != 1 && rowID==2){
                         rowID=3;
                       }
                       if(rowID==3){
                         tempcntIndex=cntIndex;
                         cntIndex>1?cntIndex=1:"";
                       }
                       setFocus(0);
                       break;
```

```
            case VK_DOWN :
                          if(rowID>1){
                            rowID--;
                            tempcntIndex!=-1?cntIndex=tempcntIndex:"";
                            tempcntIndex!=-1;
                          }
                          if(      video.playState != 1 && rowID==2){
                            rowID=1;
                          }
                          setFocus(0);
                          break
            case VK_ENTER :
                          $(".playerBottom").is(":visible")?
selectedButton():"";
                          break;
        }
        showPlayer();
        triggerHide();
    });
});
```

Implementing actions for button click(videoplayer.js)

Implements a JavaScript code for performing appropriate operation when you click on a button in the screen.

stopMedia method is used to stops the running video .

```
case 0: stopMedia();
        break;
```

playMedia method is used to play video content.

```
case 1: playMedia();
        break;
```

rewindMedia method is used to rewind the running video by 10second.

```
case 2:rewindMedia();
        break;
```

forwardMedia method is used to forward the running video by 10 second.

```
case 3: forwardMedia();
        break;
```

optionMedia method is used to open window for settings. This window is used to setup the aspect ratio for full screen video, picture quality adjustment and audio adjustment.

```
case 4: optionMedia();
        break;
```

Changing the play state of video (videoplayer.js)

If there is any change in playing state of video, this event listener method will get called.

For example,
- If play state of video is 0, video is stopped, then resetProgress method will reset all the functionalities.

- If play state of video is 5, video is stopped, then playMedia method will play video content.

- If play state of video is 1, video is playing, then get the source information like movie name and playing information like remaining and total time of currently running video.

```
function playStateChange()
{
    if (video.playState == 0 ) {
        resetProgress();
    }else if( video.playState == 5){
        resetProgress();
        vidIndex<videoArray.length-1 ?vidIndex++:vidIndex=0;
        playMedia();
    }
    else if (video.playState == 1) {
        $('#progressBall').attr('class', 'progressBall');
        triggerHide();
        getVideoSourceInfo();
        getVideoPlayInfo();
    } else if (video.playState == 6) {
        //notifyError;
    }
}
```

Buffering of video (videoplayer.js)

This method is used to implement the buffering of the video. First it will calculate buffered position and playing position of video. As video buffers, the progress bar will gradually increase with light pink color, showing the buffering progress.

setBufferPosition method is used to show the buffering progress.

```
function buffering()
{
    videoPlayInfo = video.mediaPlayInfo();
    if(videoPlayInfo)
    {
         if(videoPlayInfo.bufRemain!=-1){
            var bufferPos=
Math.ceil((videoPlayInfo.bufRemain/videoPlayInfo.duration)*progressBarWidt
h);
            var pos=videoPlayInfo.duration-videoPlayInfo.bufRemain;
            if(pos != 0)
            {
                pos=(pos/videoPlayInfo.duration)*progressBarWidth;
                bufferPos+=pos;
                setBufferPosition(bufferPos);
            }
         }
    }

}
```

Playing information of running video (videoplayer.js)

This method is used to get playing information like total time and remaining time of the currently running video.

setPosition method is used to show the running video progress, it keeps track of progress status bar & progress ball as video runs.

```
function getVideoPlayInfo() {
    videoPlayInfo = video.mediaPlayInfo();
    if (videoPlayInfo) {
        if ((video.playState != 5) && (video.playState != 0) &&
(video.playState != 2) ) {

    $("#remainingTime").text(getTimeFromMS(videoPlayInfo.currentPosition))
;
           $("#totalTime").text(" / " +
getTimeFromMS(videoPlayInfo.duration));
           var
pos=Math.ceil((videoPlayInfo.currentPosition/videoPlayInfo.duration)*progr
essBarWidth);
           setPosition(pos);
           playTimerId=setTimeout("getVideoPlayInfo()", 100);
        }
    }
}
```

Source information of running video (videoplayer.js)

This method is used to get the source information like movie name and type of running video.

```
function getVideoSourceInfo() {
    var videoSourceInfo = video.getSrcInfo();
    videoSourceInfo.title!=null?$(".runningMovieName").text(videoSourceInf
o.title):"";
}
```

Auto playing of video (videoplayer.js)

This method will set the source of video object and start the video without clicking play button.

checkFileExtensions method is used to check the supported formats i.e. mime type.

```
function loadDataSrc(auto)
{
    var vidPath=videoArray[vidIndex];
    if(checkFileExtensions(vidPath)){
        video.data ="media/"+vidPath;
        video.play(1);
    }
}
```

Removing focus from Control (videoplayer.js)

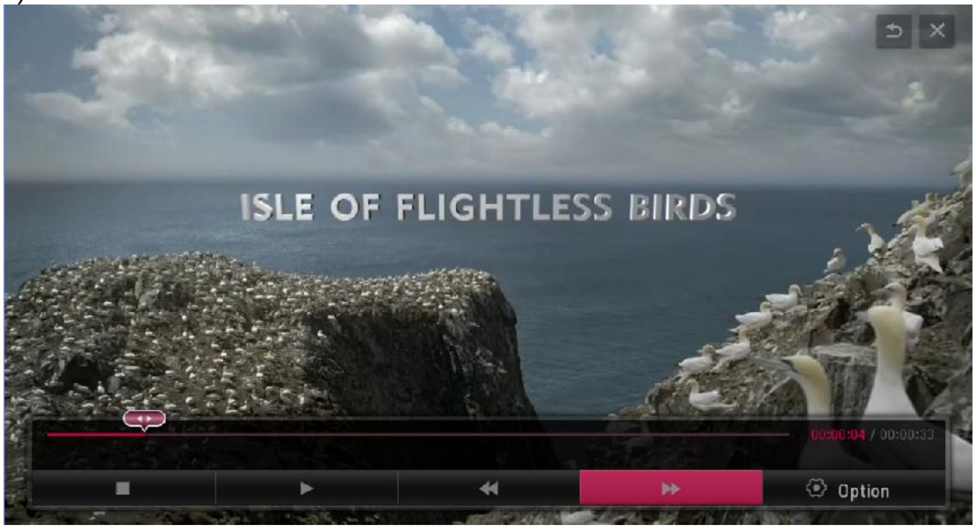
This method is used to change the color of the button from focus to normal on mouse over or keynavigation event.

```
function resetFocus(){
    $("#buttonLayout li").each(function(i){
        $(allMenuObject[i]).removeClass(defaultClass[i]+"Hover");
    });
    $(".keyHelp div").each(function(i){
        $(allExitObject[i]).removeClass(exitBtnClass[i]+"Hover");
    });
    $("#progressBall").removeClass("progressBallHover");
}
```

3.2 Expected Result

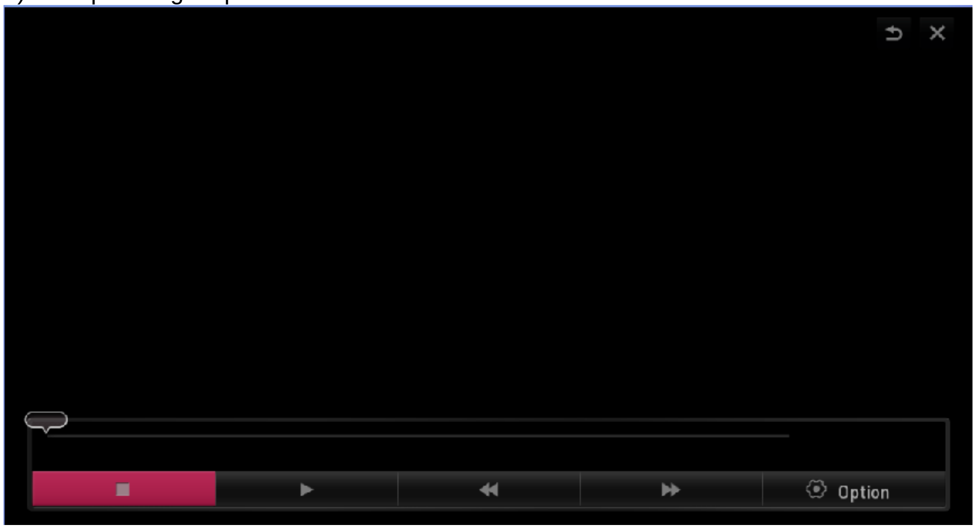
The following screen will be displayed when the sample application starts on the LG Smart TV Emulator 2012. When the application starts, the initial screen is displayed. When a user clicks a button, corresponding operation will start.

1) Initial Screen



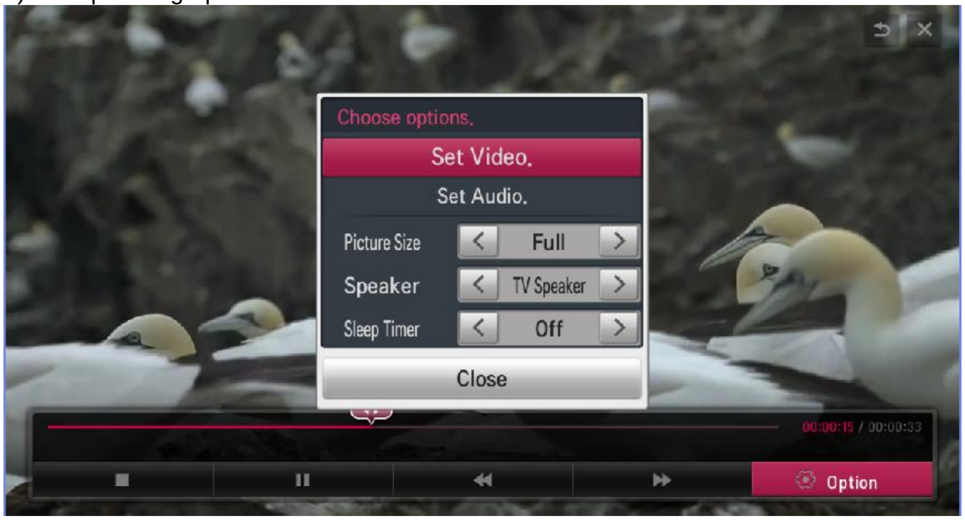
[Figure 3] WebAppSample__Media Plugin Video Player_Initial Screen

2) After pressing stop button



[Figure 4] WebAppSample__Media Plugin Video Player_Stop Screen

3) After pressing option button



Choose options.

Set Video.

Set Audio.

Picture Size: Full

Speaker: TV Speaker

Sleep Timer: Off

Close

00:00:15 / 00:00:33

Option

[Figure 5] WebAppSample__Media Plugin Video Player_Option Screen

4 Testig the Application

This chapter briefly describes how to test a developed application.

To debug the application on emulator, refer to **Testing > Testing App on Emulator** section in Developer Library in LG Developer (**http://developer.lge.com**) website

To run and debug the application on real TV, refer to **Testing > Deploying and Testing App on Real TV** section in Developer Library in LG Developer website.