LG Electronics

[Tutorial] LG Web_Controlling TV Channel in Broadcast Application

Version 1.2 – October 2012

LGDEV-064

Home Entertainment Company
LG Electronics, Inc.

Copyright

About This Document

Revision History

Document Version	Date	Comment
1.2	October 11, 2012	Needed APIs and Sample codes are updated. Section 2.2, 2.3, 2.4, 2.5 are added.
1.1	February 13, 2012	Needed APIs and Sample codes are updated.
1.0	December 14, 2011	Initial Version

Purpose

This document describes how to control TV channel in broadcast application using LG Web Open APIs.

Reference Documents

Refer to the following documents:
- LG Web Application Development Guide
- LG Web Open API Reference Guide

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
API	Application Programming Interface

Contents

Tables

Figures

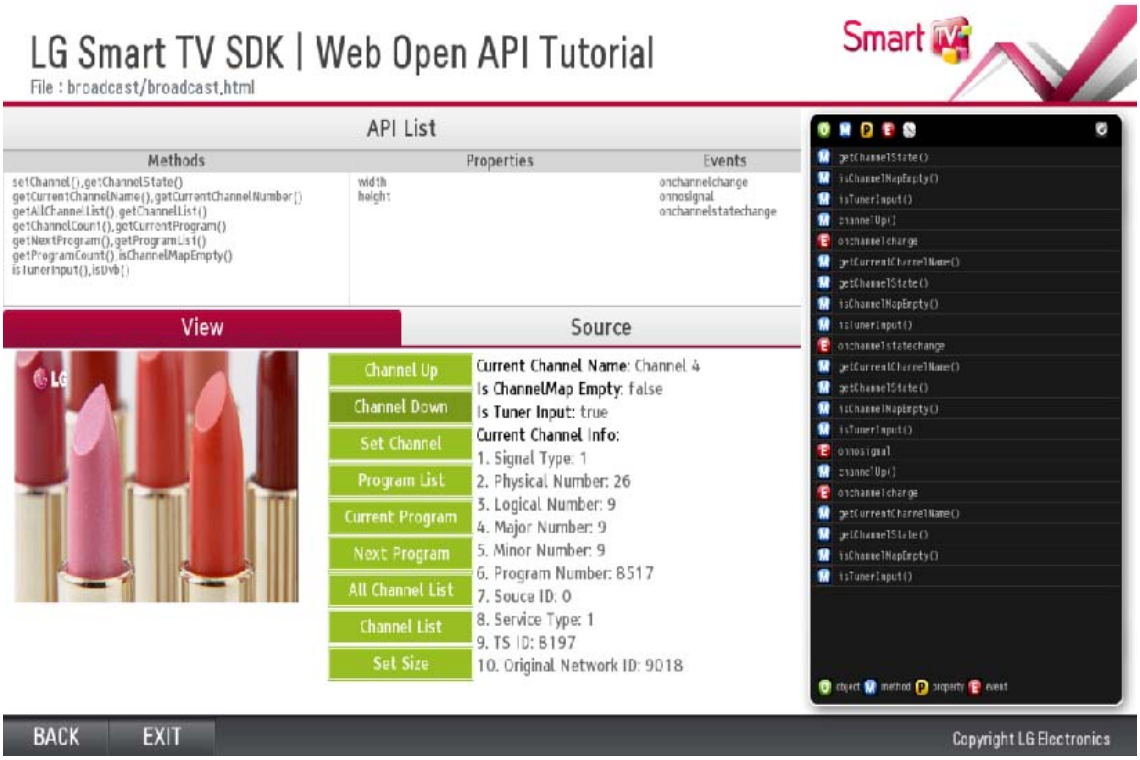
1 Introduction

This chapter provides an overview of this application and needed APIs.

1.1 Overview

This application is designed to show which method, properties, and events of LG Web Open API are used for displaying TV broadcast images in the application of LG Smart TV.

This application shows how to use broadcast API for changing channels(by using up and down key), setting channels, getting list of program running on current channel, getting information of current and next program running on current channel, list of available channels on LG Smart TV, and many more. This application also shows list of events occurred while changing channel or when there is no signal. It will also show how to change properties(width and height) of broadcasted image.



[Figure 1] Web Application using Broadcast API

1.2 Needed APIs

This application uses following Web Open API:

[Table 1] Description of the Needed APIs

API Class	Name	Description
Method	channelUp	Change channel using Up key of remote.
	channelDown	Change channel using Down key of remote.
	getCurrentChannelName	Return name of current channel.
	getCurrentProgram	Used to view the info of the program broadcasted on the current channel.
	getNextProgram	Used to view info of the program which are coming after current program broadcasted on the current channel.
	getCurrentChannelNumber	Used to view the physical number, major number, minor number, program number, source ID and TSID (Transport Stream ID) of the current channel.
	getProgramList(startTime, endTime)	Used to get date and program information of the current channel.
	getProgramCount	Used to get the number of programs for the specified date of current channel.
	setChannel (signalType, physicalNum, logicalNum)	Used to move directly to the specified channel.
	getChannelList(serviceType,signalType)	Used to get the information of searchable channels of the current TV. Based on index, maximally 10 result values are returned.
	getChannelCount(serviceType,inputType)	Used to get the number of searchable channels of the current TV.
	getChannelState	Used to get status of TV signal for current program.
	isChannelMapEmpty	Used to determine whether there is a channel map of the current TV.
	isTunerInput	Used to determine whether the current input signal is from TV.
	isDvb	Used to determine whether the TV signal complies with DVB or ATSC standard.
Property	width	return the width information of the broadcast object as string types.
	height	return the height information of the broadcast object as string types.
Event	onchannelchange	This event occurs when a TV channel is changed.
	onnosignal	This event occurs when there is no signal of the TV tuner.
	onchannelstatechange	This event occurs when TV signal status is changed.

For more information on these functions, refer to “LG Web Open API Reference Guide”.

Note

Log is used for checking the sequence of Web Open API; this will not be covered in the sample code description.

2 Creating Application

This chapter describes how to control TV channel in broadcast application using LG Web Open APIs.

2.1 Initializing the Page

Use the **initPage** function to set the basic functions of the application.

04: Record the last visited page when running the application.
07: Initialize the page.
08: Get the source code of the page using the XMLHttpRequest object.
09: Initialize the Log function.
11-12: Initialize commonly used broadcast and msgArea variable .
14: Check NetCast version.
17-18: Add event handlers which will be executed when the corresponding button is pressed.
19-20: Add event handlers related to broadcast.
22: Print Broadcast object to blackbird console using `lgobject` function.
24: Call function to initialize broadcast.
25: Call function to show property values.

Sample Code

```
01 : function initPage(){
02 :
03 :   //save page as last visited page
04 :         setLastVisitPage();
05 :
06 :         //common initialize function
07 :         commonInitialize();
08 :         requestSourceCode();
09 :         jsLog.initLG();
10 :
11 :         broadcast = document.getElementById('broadcast');
12 :         msgArea = document.getElementById('displayArea');
13 :
14 :         checkNetCastVersion();
15 :
16 :         //add onclick event handler
17 :         addEventHandler(document.getElementById("btn_back"),     "click",
     onClickHandler);
18 :         addEventHandler(document.getElementById("btn_exit"),     "click",
     onClickHandler);
19 :         addEventHandler(document.getElementById("channel_up"),   "click",
     channelUpHandler);
20 :         addEventHandler(document.getElementById("channel_down"),
     "click", channelDownHandler);
21 :
22 :         jsLog.lgobject('application/x-netcast-broadcast');
23 :
24 :         loadContent();
25 :         showPropResult()
26 : }
```

2.2 Check NetCast Version

Use **checkNetCastVersion** function to check the NetCast version and APIs are supported on NetCast 3.0 1st SU or later.

02: Declare nBrowserVersion.
05-11: If return value of nBrowserVersion is greater than equal to five then add event handler for following button: Set channel, Program list, Current Program Info, Next Program Info, All channel list, Channel List and set Size.
13: Add listener for channel change event.
14: Add listener for event when there is no signal of the TV tuner.
15: Add listener for event when TV signal status is changed.
17-21: Display Methods, properties and events of broadcast into respective div.

Sample Code

```
01 : function checkNetCastVersion(){
02 :   var nBrowserVersion = getBrowserVersion();
03 :   if(nBrowserVersion >= 5) // NetCast 3.0
04 :   {
05 :                 addEventHandler(document.getElementById("set_channel"),
     "click", setChannelHandler);
06 :                 addEventHandler(document.getElementById("program_list"),
     "click", programListHandler);
07 :
       addEventHandler(document.getElementById("current_programInfo"),
     "click", currentProgramInfoHandler);
08 :
       addEventHandler(document.getElementById('next_programInfo'),    "click",
     nextProgramInfoHandler);
09 :
       addEventHandler(document.getElementById("all_channelList"),     "click",
     allChannelListHandler);
10 :                 addEventHandler(document.getElementById("channelList"),
     "click", channelListHandler);
11 :                 addEventHandler(document.getElementById("set_size"),
     "click", setSizeHandler);
12 :
13 :                 broadcast.onchannelchange  =  processChannelChangeFunction;

14 :                 broadcast.onnosignal = processNoSignalFunction;
15 :
       broadcast.onchannelstatechange=processChStateChangeFunction;
16 :
17 :                 setInnerTextById("method",
     "setChannel(),getChannelState()<br>getCurrentChannelName(),
     getCurrentChannelNumber()<br>getAllChannelList(),getChannelList()
18 : <br>getChannelCount(),getCurrentProgram()<br>getNextProgram(),getProgramL
     ist()
19 : <br>getProgramCount(),isChannelMapEmpty()<br>isTunerInput(),isDvb()");
20 :                 setInnerTextById("property", "width<br>height");
21 :                 setInnerTextById("events",
     "onchannelchange<br>onnosignal<br>onchannelstatechange");
22 :         }
23 : }
```

2.3 Handling Button Click Events

The following functions are event handlers which were added in **initPage** function.

channelUpHandler
This function is called when ‘channel up’ button is clicked.

03: Change the channel number to one higher than current channel.

channelDownHandler
This function is called when ‘channel down’ button is clicked.

09: Change the channel number to one lower than current channel.

Sample Code

```
01 :    function channelUpHandler() {
02 :
03 :                broadcast.channelUp();
04 :                jsLog.lgmethod('channelUp()');
05 :         }
06 :
07 :    function channelDownHandler() {
08 :
09 :                broadcast.channelDown();
10 :                jsLog.lgmethod('channelDown()');
11 :    }
```

The following functions are event handlers which were added in **checkNetCastVersion** function.

setChannelHandler
This function is called when ‘Set Channel’ button is clicked.

12: Get the information of searchable channels of the current TV for DIGITAL signal.
15: If channel list is undefined get the information of searchable channels of the current TV for ANALOG signal.
19-21: Find the channel name, signal type and physical number from channelList.
22-25: If TV signal type is DVB then define logical number and call setChannel to change the channel directly to the specified channel.
27-30: If signal type is ATSC then define major number and minor number and call setChannel to change the channel directly to the specified channel.

Sample Code

```
 01 :   function setChannelHandler() {
 02 :              var channelList;
 03 :              var channelCount;
 04 :              var channelName;
 05 :              var signalType;
 06 :              var physicalNum;
 07 :              var majorNum;
 08 :              var minorNum;
 09 :              var logicalNum;
 10 :              var channelToSet = 2;
 11 :
 12 :              channelList      =      broadcast.getAllChannelList("tv",
    "DIGITAL");
 13 :              jsLog.lgmethod("getAllChannelList()")
 14 :              if (channelList == undefined) {
 15 :                    channelList   =   broadcast.getAllChannelList("tv",
    "ANALOG");
 16 :                    jsLog.lgmethod("getAllChannelList()")
 17 :              }
 18 :
 19 :              channelName = channelList[channelToSet].channelName;
```

```
20 :                signalType = channelList[channelToSet].signalType;
21 :                physicalNum = channelList[channelToSet].physicalNum;
22 :                if (broadcast.isDvb()) {
23 :                      logicalNum  =  channelList[channelToSet].logicalNum;
   // For DVB,
24 :                      broadcast.setChannel(signalType,       physicalNum,
   logicalNum);
25 :                      jsLog.lgmethod("setChannel()")
26 :                } else {
27 :                majorNum  =  channelList[channelToSet].majorNum;  //  For
   ATSC,
28 :                minorNum = channelList[channelToSet].minorNum;
29 :                broadcast.setChannel(signalType,  physicalNum,  majorNum,
   minorNum);
30 :                      jsLog.lgmethod("setChannel()")
31 :                }
     }
```

programListHandler
This function is called when 'Program List' button is clicked.

03-11: Calculating the value of current date and end date parameter in required format i.e. YYYYMMDDHHmmss.
13: Get the number of programs for the specified date of current channel.
15: Display the program count value by using htmlStr1.
16: Get date and program information of the current channel in programList variable.
17-20: Define information like title, start time, end time, and description of the program using programList variable
31-33: Display all the information by using htmlStr1 and display it by calling showPropResult function.
38-43: Modifying date value to respective text string.

Sample Code

```
01 :   function programListHandler() {
02 :
03 :                var today = new Date();
04 :                var tYear = today.getYear() + 1900;
05 :                var tMonth = modNumberToText(today.getMonth() + 1);
06 :                var tDate = modNumberToText(today.getDate());
07 :                var tHour = modNumberToText(today.getHours());
08 :                var tMin = modNumberToText(today.getMinutes());
09 :                var tSec = modNumberToText(today.getSeconds());
10 :                var start_date = tYear + tMonth + tDate + tHour + tMin +
   tSec; //YYYYMMDDHHmmss
11 :                var end_date = tYear + tMonth + tDate + "235959";
   //YYYYMMDDHHmmss
12 :
13 :                var programCount = broadcast.getProgramCount(start_date,
   end_date);
14 :                jsLog.lgmethod('getProgramCount()');
15 :                htmlStr1 += "<font style='color:#000000; font-
   weight:bold'>All Program Count: </font>" + programCount + "<br>";
16 :                var programList = broadcast.getProgramList(start_date,
   end_date);
17 :                jsLog.lgmethod('getProgramList()');
18 :                var title = new Array();
19 :                var startTime = new Array();
20 :                var endTime = new Array();
21 :                var description = new Array();
22 :
23 :                if (programCount != 0) {
24 :                      htmlStr1 += "Program List: <br>";
25 :
26 :                      for (i = 0; i < programList.length; i++) {
27 :                            title[i] = programList[i].title;
28 :                            startTime[i] = programList[i].startTime;
29 :                            endTime[i] = programList[i].endTime;
30 :                            description[i] = programList[i].description;
31 :                            htmlStr1 += "[" + i + "] Title: " + title[i]
```

```
    + "<br>Start Time: " + startTime[i] + "<br>End Time: " + endTime[i] +
    "<br>Description: " + description[i] + " <br>";
32 :                            msgArea.innerHTML = htmlStr1 + "\n";
33 :                        }
34 :                        showPropResult();
35 :                }
36 :
37 :        }
38 :    function modNumberToText(num) {
39 :
40 :                if (num < 10) num = "0" + num;
41 :
42 :                return num;
43 :        }
```

currentProgramInfoHandler
This function is called when 'Current Program' button is clicked.

02: Get value of title, start time, end time, description and event ID of the program broadcasted on the current channel.
04-05: Display all the information by using htmlStr1 and display it by calling showPropResult function.

Sample Code

```
01 :    function currentProgramInfoHandler() {
02 :                var currentProgram = broadcast.getCurrentProgram();
03 :                jsLog.lgmethod('getCurrentProgram()');
04 :                htmlStr1 += "Title: " + currentProgram.title + "<br>Start
    Time: " + currentProgram.startTime + "<br>End Time: " +
    currentProgram.endTime + "<br> Description: " +
    currentProgram.description + "<br> Event Id: " + currentProgram.eventId
    + " ";
05 :                showPropResult();
06 :        }
```

nextProgramInfoHandler
This function is called when 'Next Program' button is clicked.

02: Get the title, start time, end time, description and event ID of the program which are coming after current program broadcasted on the current channel.
04-05: Display all the information by using htmlStr1 and display it by calling showPropResult function.

Sample Code

```
01 :    function nextProgramInfoHandler() {
02 :                var nextProgram = broadcast.getNextProgram();
03 :                jsLog.lgmethod('getNextProgram()');
04 :                htmlStr1 += "Title: " + nextProgram.title + "<br>Start
    Time: " + nextProgram.startTime + "<br>End Time: " + nextProgram.endTime
    + "<br> description: " + nextProgram.description + "<br> Event Id: " +
    nextProgram.eventId + " ";
05 :                showPropResult();
06 :
07 :        }
```

allChannelListHandler
This function is called when 'All Channel List' button is clicked.

04-09: Define parameters of broadcast object.
10-12: Get the number of searchable channels of the current TV for all signal type, Digital and Analog signal type
13: Assign all the channel count values in htmlStr1.
15: Get the information of searchable channels of the current TV for DIGITAL signal.
18-19: If channel list is undefined for DIGITAL type, display no digital channel using htmlStr1.
22-26: If channel list is not undefined get channel name, signal type and physical number from

channelList.

27-30: If TV signal type is DVB then define logical number from channelList and assign all parameter to htmlStr1.

32-34: If signal type is ATSC then define major number and minor number from channelList and assign all parameter to htmlStr1.

37-40: Assign htmlStr1 value to msgArea div and display it by calling showPropResult function.

Sample Code

```
01 :   function allChannelListHandler() {
02 :               var channelList;
03 :               var channelCount;
04 :               var channelName = new Array();
05 :               var signalType = new Array();
06 :               var physicalNum = new Array();
07 :               var majorNum = new Array();
08 :               var minorNum = new Array();
09 :               var logicalNum = new Array();
10 :               var channelCount = broadcast.getChannelCount("tv", "all");
11 :               var digitalChCount = broadcast.getChannelCount("tv",
    "DIGITAL");
12 :               var analogChCount = broadcast.getChannelCount("tv",
    "ANALOG");
13 :               htmlStr1 += "<font style='color:#000000; font-
    weight:bold'>All Channel Count:</font> " + channelCount + " [Digital: "
    + digitalChCount + ", Analog: " + analogChCount + "]";
14 :               jsLog.lgmethod('getChannelCount()');
15 :               channelList = broadcast.getAllChannelList("tv",
    "DIGITAL");
16 :               jsLog.lgmethod("getAllChannelList()")
17 :
18 :               if (channelList == undefined) {
19 :                     htmlStr1 += "<br>No Digital Channel<br>";
20 :
21 :               } else {
22 :                     htmlStr1 += "<br> List of Digital Channel :";
23 :                     for (i = 0; i < channelList.length; i++) {
24 :                           channelName[i] = channelList[i].channelName;
25 :                           signalType[i] = channelList[i].signalType;
26 :                           physicalNum[i] = channelList[i].physicalNum;
27 :                           if (broadcast.isDvb()) {
28 :                                 jsLog.lgmethod('isDvb()');
29 :                                 logicalNum[i] =
    channelList[i].logicalNum; // For DVB
30 :                                 htmlStr1 += "<br> [" + (i + 1) + "]
    Channel Name: " + channelName[i] + "<br>Signal Type: " + signalType[i] +
    "<br>Physical Number: " + physicalNum[i] + "<br> Logical Number: " +
    logicalNum[i] + " ";
31 :                           } else {
32 :                     majorNum[i] = channelList[i].majorNum; // For ATSC
33 :                     minorNum[i] = channelList[i].minorNum;
34 :                                 htmlStr1 += "<br> [" + (i + 1) + "]
    Channel Name: " + channelName[i] + "<br> Signal Type: " + signalType[i]
    + "<br> Physical Number: " + physicalNum[i] + "<br>Major Number: " +
    majorNum[i] + "<br> Minor Number: " + minorNum[i] + " ";
35 :                           }
36 :                           msgArea.innerHTML = htmlStr1 + "\n";
37 :                     }
38 :               }
39 :               showPropResult();
40 :   }
```

channelListHandler

This function is called when ‘Channel List’ button is clicked.

04-10: Define parameters of broadcast object.

11-13 Get the number of searchable channels of the current TV for all signal type, Digital and Analog signal type

14: Assign all the channel count values in htmlStr1.

16: Get the information of searchable channels of the current TV for DIGITAL signal, starting from 1st index. Based on index, maximally 10 result values are returned.
12-20: If channel list is undefined for DIGITAL type, display no digital channel using htmlStr1.
22-27: If channel list is not undefined get channel name, signal type and physical number from channelList.
28-31: If TV signal type is DVB then define logical number from channelList and assign all parameter to htmlStr1.
32-35: If signal type is ATSC then define major number and minor number from channelList and assign all parameter to htmlStr1.
37-40: Assign htmlStr1 value to msgArea div and display it by calling showPropResult function.

Sample Code

```
01 :   function channelListHandler() {
02 :
03 :               var channelList;
04 :               var channelCount;
05 :               var channelName = new Array();
06 :               var signalType = new Array();
07 :               var physicalNum = new Array();
08 :               var majorNum = new Array();
09 :               var minorNum = new Array();
10 :               var logicalNum = new Array();
11 :               var channelCount = broadcast.getChannelCount("tv", "all");
12 :               var digitalChCount = broadcast.getChannelCount("tv",
    "DIGITAL");
13 :               var analogChCount = broadcast.getChannelCount("tv",
    "ANALOG");
14 :               htmlStr1 += "<font style='color:#000000; font-
    weight:bold'>All Channel Count:</font> " + channelCount + " [Digital: "
    + digitalChCount + ", Analog: " + analogChCount + "]";
15 :               jsLog.lgmethod('getChannelCount()');
16 :               channelList = broadcast.getChannelList("tv", "DIGITAL",
    1);
17 :               jsLog.lgmethod("getChannelList()")
18 :
19 :               if (channelList == undefined) {
20 :                     htmlStr1 += "<br>No Digital Channel<br>";
21 :
22 :               } else {
23 :                     htmlStr1 += "<br> List of Digital Channel :";
24 :                     for (i = 0; i < channelList.length; i++) {
25 :                           channelName[i] = channelList[i].channelName;
26 :                           signalType[i] = channelList[i].signalType;
27 :                           physicalNum[i] = channelList[i].physicalNum;
28 :                           if (broadcast.isDvb()) {
29 :                                 jsLog.lgmethod('isDvb()');
30 :                                 logicalNum[i] =
    channelList[i].logicalNum; // For DVB
31 :                                 htmlStr1 += "<br> [" + (i + 1) + "]
    Channel Name: " + channelName[i] + "<br>Signal Type: " + signalType[i] +
    "<br>Physical Number: " + physicalNum[i] + "<br> Logical Number: " +
    logicalNum[i] + " ";
32 :                           } else {
33 :                     majorNum[i] = channelList[i].majorNum; // For ATSC
34 :                     minorNum[i] = channelList[i].minorNum;
35 :                                 htmlStr1 += "<br> [" + (i + 1) + "]
    Channel Name: " + channelName[i] + "<br> Signal Type: " + signalType[i]
    + "<br> Physical Number: " + physicalNum[i] + "<br>Major Number: " +
    majorNum[i] + "<br> Minor Number: " + minorNum[i] + " ";
36 :                           }
37 :                           msgArea.innerHTML = htmlStr1 + "\n";
38 :                     }
39 :               }
40 :               showPropResult();
41 :         }
```

setSizeHandler
This function is called when ‘Set Size’ button is clicked.

02: Change the broadcast properties i.e. width and height as per current value.

Sample Code

```
01 :   function setSizeHandler() {
02 :               if ((broadcast.width == 240) && (broadcast.height == 180))
    {
03 :                     broadcast.width = 320;
04 :                     broadcast.height = 240;
05 :               } else {
06 :                     broadcast.width = 240;
07 :                     broadcast.height = 180;
08 :               }
09 :               jsLog.lgproperty('width:' + broadcast.width);
10 :               jsLog.lgproperty('height:' + broadcast.height);
11 :         }
```

2.4 Handling Broadcast Events

The following functions are broadcast event handlers which were added in checkNetCastVersion function.

processChannelChangeFunction
This function is handler of event **onchannelchange** which occur when a TV channel is changed.

03: If event occur display all the information by using loadContent and showPropResult function.

processChStateChangeFunction
This function is handler of **onchannelstatechange** event. This event occurs when TV signal status is changed.

12-14: If event occur display all the information by using loadContent and showPropResult function

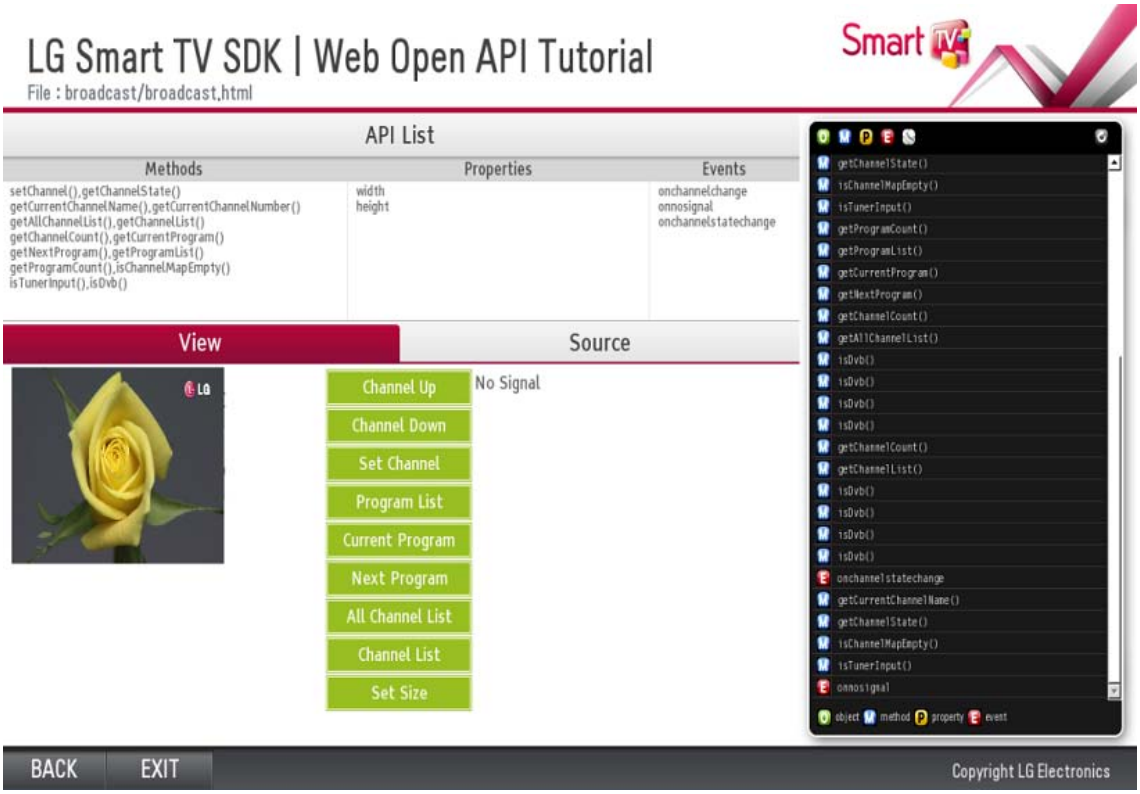
processNoSignalFunction
This function is handler of **onnosignal** event. This event occurs when there is no signal of the TV tuner.

19-21: If event occur display NO Signal by using htmlStr.
22: If event does not occur display htmlStr in msgArea div.

Sample Code

```
01 :   function processChannelChangeFunction(e) {
02 :                if (e) {
03 :                       jsLog.lgevent('onchannelchange');
04 :                       loadContent();
05 :                       showPropResult();
06 :                }
07 :         }
08 :
09 :   function processChStateChangeFunction(e) {
10 :                if (e) {
11 :                       jsLog.lgevent('onchannelstatechange');
12 :                       loadContent();
13 :                       showPropResult();
14 :                }
15 :         }

16 :   function processNoSignalFunction(e) {
17 :                var htmlStr = "";
18 :                if (e) {
19 :                       htmlStr += "No Signal";
20 :                       jsLog.lgevent('onnosignal');
21 :                }
22 :                msgArea.innerHTML = htmlStr;
23 :         }
```



[Figure 2] Displaying No Signal Event of Broadcast

2.5 Loading Content

The following code is used to display the broadcast object information on screen.

loadContent

03: If current channel name is not undefined then display the current channel name using htmlStr1.
05: If current channel status is not undefined then display the status TV signal for current program using htmlStr1.
07: If current channel name is not undefined then check whether there is a channel map of the current TV.
09: If current channel name is not undefined then check whether the current input signal is from TV.
12-22: Define the value of currentChannel and display physical number, major number, minor number, program number, source ID, transport stream ID and original network ID of the current channel.
23: If the value of currentChannel is not undefined then append all the above value to htmlStr1.

showPropResult

26: Assign the value of htmlStr1 to msgArea div.
27: Set the overflow style of msgArea div to auto.
28: Reinitialize the htmlStr1 value.

Sample Code

```
01 :   function loadContent() {
02 :
03 :                htmlStr1    =    (broadcast.getCurrentChannelName()    !=
   undefined)   ?   "<font   style='color:#000000;   font-weight:bold'>Current
   Channel Name:</font> " + broadcast.getCurrentChannelName() : "";
04 :                jsLog.lgmethod('getCurrentChannelName()');
05 :                htmlStr1 += (broadcast.getChannelState() != undefined) ?
   "<br><font style='color:#000000; font-weight:bold'>Channel State:</font> "
   + broadcast.getChannelState() : "";
06 :                jsLog.lgmethod('getChannelState()');
07 :                htmlStr1    +=    (broadcast.getCurrentChannelName()    !=
   undefined)   ?   "<br><font   style='color:#000000;   font-weight:bold'>Is
   ChannelMap Empty:</font> " + broadcast.isChannelMapEmpty() : "";
08 :                jsLog.lgmethod('isChannelMapEmpty()');
09 :                htmlStr1    +=    (broadcast.getCurrentChannelName()    !=
   undefined) ? "<br><font style='color:#000000; font-weight:bold'>Is Tuner
   Input:</font> " + broadcast.isTunerInput() : "";
10 :                jsLog.lgmethod('isTunerInput()');
11 :
12 :                var currentChannel = broadcast.getCurrentChannelNumber();
13 :                var signalType = currentChannel.signalType;
14 :                var physicalNum = currentChannel.physicalNum;
15 :                var logicalNum = currentChannel.logicalNum;
16 :                var minorNum = currentChannel.minorNum;
17 :                var majorNum = currentChannel.majorNum;
18 :                var programNum = currentChannel.programNum;
19 :                var sourceId = currentChannel.sourceId;
20 :                var serviceType = currentChannel.serviceType;
21 :                var tsId = currentChannel.tsId;
22 :                var nwId = currentChannel.originalNetworkId;
23 :                htmlStr1 += (currentChannel != undefined) ? "<br><font
   style='color:#000000; font-weight:bold'>Current Channel Info:</font><br>1.
   Signal Type: " + signalType + "<br>2. Physical Number: " + physicalNum +
   "<br>3. Logical Number: " + logicalNum + "<br>4. Major Number: " +
   majorNum + "<br>5. Minor Number: " + minorNum + "<br>6. Program Number: "
   + programNum + "<br>7. Souce ID: " + sourceId + "<br>8. Service Type: " +
   serviceType + "<br>9. TS ID: " + tsId + "<br>10. Original Network ID: " +
   nwId : "";
24 :   }
25 :   function showPropResult() {
26 :                msgArea.innerHTML = htmlStr1;
27 :                msgArea.style.overflow = "auto";
28 :                htmlStr1 = "";
29 :         }
```

2.6 Inputting Keys

Use **onUserInput** function is called by onClickHandler function; it receives a key value as the userInput parameter from onClickHandler and creates the corresponding function for each key value to operate the key.

05: When the back key is pressed, it will open previous page i.e. main menu page.

Sample Code

```
01 :   function onUserInput(userInput) {
02 :   switch (userInput) {
03 :   case VK_BACK :
      window.location.replace(“../main_menu.html”);
04 :   break;
05 :          }
06 :   }
```

2.7 Setting Broadcast Object

The following code shows how to set voice recognition object.

01: Set id value as ‘broadcast’.
02: Set data type as ‘`application/x-netcast-broadcast`’.
03: Set width and height property of broadcast.

Sample Code

```
01 : <object id="broadcast"
02 :  type="application/x-netcast-broadcast"
03 :  width=240 height=180>
04 : </object>
```

2.8 Source Code of broadcast.html

Source code of broadcast.html is as follows:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>

<head>
<meta http-equiv="Content-Type" content="text/html;  charset=UTF-8">
<title>Broadcast Api Test</title>
<link rel="stylesheet" href="../css/style.css">
<script language="javascript" src="../js/common.js"></script>
<script language="javascript" src="../js/keycode.js"></script>
<script language="javascript" src="../js/menu.js"></script>
<script type="text/javascript" src="../blackbirdjs/blackbird.js"></script>
<link type="text/css" rel="Stylesheet" href="../blackbirdjs/blackbird.css"
/>
<script>
   addEventHandler(window, "load", initPage);
   var htmlStr1 = "";
   var broadcast;
   var msgArea;
   //initialize page
   function initPage() {

      //save page as last visited page
      setLastVisitPage();

      //common initialize function
      commonInitialize();
      equestSourceCode();
      jsLog.initLG();

      broadcast = document.getElementById('broadcast');
      msgArea = document.getElementById('displayArea');

      checkNetCastVersion();

      //add onclick event handler
      addEventHandler(document.getElementById("btn_back"), "click", onClickHandler);
      addEventHandler(document.getElementById("btn_exit"), "click", onClickHandler);
      addEventHandler(document.getElementById("channel_up"), "click", channelUpHandler);
      addEventHandler(document.getElementById("channel_down"), "click", channelDownHandler);

      jsLog.lgobject('application/x-netcast-broadcast');
```

```
      loadContent();
      showPropResult()
   }

   function processChannelChangeFunction(e) {
      if (e) {
         jsLog.lgevent('onchannelchange');
         loadContent();
         showPropResult();
      }
   }

   function processChStateChangeFunction(e) {
      if (e) {
         sLog.lgevent('onchannelstatechange');
         loadContent();
         showPropResult();
      }
   }

   function loadContent() {
      htmlStr1 = (broadcast.getCurrentChannelName() != undefined) ? "<font style='color:#000000; font-weight:bold'>Current Channel
Name:</font> " + broadcast.getCurrentChannelName() : "";
      jsLog.lgmethod('getCurrentChannelName()');
      htmlStr1 += (broadcast.getChannelState() != undefined) ? "<br><font style='color:#000000; font-weight:bold'>Channel
State:</font> " + broadcast.getChannelState() : "";
      jsLog.lgmethod('getChannelState()');
      htmlStr1 += (broadcast.getCurrentChannelName() != undefined) ? "<br><font style='color:#000000; font-weight:bold'>Is ChannelMap
Empty:</font> " + broadcast.isChannelMapEmpty() : "";
      jsLog.lgmethod('isChannelMapEmpty()');
      htmlStr1 += (broadcast.getCurrentChannelName() != undefined) ? "<br><font style='color:#000000; font-weight:bold'>Is Tuner
Input:
</font> " + broadcast.isTunerInput() : "";
      jsLog.lgmethod('isTunerInput()');

      var currentChannel = broadcast.getCurrentChannelNumber();
      var signalType = currentChannel.signalType;
      var physicalNum = currentChannel.physicalNum;
      var logicalNum = currentChannel.logicalNum;
      var minorNum = currentChannel.minorNum;
      var majorNum = currentChannel.majorNum;
      var programNum = currentChannel.programNum;
      var sourceId = currentChannel.sourceId;
      var serviceType = currentChannel.serviceType;
```

```
      var tsId = currentChannel.tsId;
      var nwId = currentChannel.originalNetworkId;
      htmlStr1 += (currentChannel != undefined) ? "<br><font style='color:#000000; font-weight:bold'>Current Channel
Info:</font><br>1. Signal Type: " + signalType + "<br>2. Physical Number: " + physicalNum + "<br>3. Logical Number: " + logicalNum +
"<br>4. Major Number: " + majorNum + "<br>5. Minor Number: " + minorNum + "<br>6. Program Number: " + programNum + "<br>7. Souce ID:
" + sourceId + "<br>8. Service Type: " + serviceType + "<br>9. TS ID: " + tsId + "<br>10. Original Network ID: " + nwId : "";

   }

   function showPropResult() {
      msgArea.innerHTML = htmlStr1;
      msgArea.style.overflow = "auto";
      htmlStr1 = "";
   }

   function processNoSignalFunction(e) {
      var htmlStr = "";
      if (e) {
         htmlStr += "No Signal";
         jsLog.lgevent('onnosignal');
      }
      msgArea.innerHTML = htmlStr;
   }
      //onUserInput function should be implemented
   function onUserInput(userInput) {

   switch (userInput) {
      case VK_BACK:
         window.location.replace("../main_menu.html");
         break;

      }
   }

  function channelUpHandler() {

      broadcast.channelUp();
      jsLog.lgmethod('channelUp()');
   }

   function channelDownHandler() {

      broadcast.channelDown();
     jsLog.lgmethod('channelDown()');
   }
```

```
   function allChannelListHandler() {
      var channelList;
      var channelCount;
      var channelName = new Array();
      var signalType = new Array();
      var physicalNum = new Array();
      var majorNum = new Array();
      var minorNum = new Array();
      var logicalNum = new Array();
      var channelCount = broadcast.getChannelCount("tv", "all");
      var digitalChCount = broadcast.getChannelCount("tv", "DIGITAL");
      var analogChCount = broadcast.getChannelCount("tv", "ANALOG");
      htmlStr1 += "<font style='color:#000000; font-weight:bold'>All Channel Count:</font> " + channelCount + " [Digital: " +
digitalChCount + ", Analog: " + analogChCount + "]";
      jsLog.lgmethod('getChannelCount()');
      channelList = broadcast.getAllChannelList("tv", "DIGITAL");
      jsLog.lgmethod("getAllChannelList()")

      if (channelList == undefined) {
         htmlStr1 += "<br>No Digital Channel<br>";

      }
      else {
         htmlStr1 += "<br> List of Digital Channel :";
         for (i = 0; i < channelList.length; i++) {
            channelName[i] = channelList[i].channelName;
            signalType[i] = channelList[i].signalType;
            physicalNum[i] = channelList[i].physicalNum;
            if (broadcast.isDvb()) {
               jsLog.lgmethod('isDvb()');
               logicalNum[i] = channelList[i].logicalNum; // For DVB
               htmlStr1 += "<br> [" + (i + 1) + "] Channel Name: " + channelName[i] + "<br>Signal Type: " + signalType[i] +
"<br>Physical Number: " + physicalNum[i] + "<br> Logical Number: " + logicalNum[i] + " ";
            }
            else {
               majorNum[i] = channelList[i].majorNum; // For ATSC
               minorNum[i] = channelList[i].minorNum;
               htmlStr1 += "<br> [" + (i + 1) + "] Channel Name: " + channelName[i] + "<br> Signal Type: " + signalType[i] + "<br>
Physical Number: " + physicalNum[i] + "<br>Major Number: " + majorNum[i] + "<br> Minor Number: " + minorNum[i] + " ";
            }
            msgArea.innerHTML = htmlStr1 + "\n";
         }
      }
      showPropResult();
```

```
    }

    function channelListHandler() {

       var channelList;
       var channelCount;
       var channelName = new Array();
       var signalType = new Array();
       var physicalNum = new Array();
       var majorNum = new Array();
       var minorNum = new Array();
       var logicalNum = new Array();
       var channelCount = broadcast.getChannelCount("tv", "all");
       var digitalChCount = broadcast.getChannelCount("tv", "DIGITAL");
       var analogChCount = broadcast.getChannelCount("tv", "ANALOG");
       htmlStr1 += "<font style='color:#000000; font-weight:bold'>All Channel Count:</font> " + channelCount + " [Digital: " +
digitalChCount + ", Analog: " + analogChCount + "]";
       jsLog.lgmethod('getChannelCount()');
       channelList = broadcast.getChannelList("tv", "DIGITAL", 1);
       jsLog.lgmethod("getChannelList()")

       if (channelList == undefined) {
          htmlStr1 += "<br>No Digital Channel<br>";

       }
       else {
          htmlStr1 += "<br> List of Digital Channel :";
          for (i = 0; i < channelList.length; i++) {
             channelName[i] = channelList[i].channelName;
             signalType[i] = channelList[i].signalType;
             physicalNum[i] = channelList[i].physicalNum;
             if (broadcast.isDvb()) {
                jsLog.lgmethod('isDvb()');
                logicalNum[i] = channelList[i].logicalNum; // For DVB
                htmlStr1 += "<br> [" + (i + 1) + "] Channel Name: " + channelName[i] + "<br>Signal Type: " + signalType[i] +
"<br>Physical Number: " + physicalNum[i] + "<br> Logical Number: " + logicalNum[i] + " ";
             }
             else {
                majorNum[i] = channelList[i].majorNum; // For ATSC
                minorNum[i] = channelList[i].minorNum;
                htmlStr1 += "<br> [" + (i + 1) + "] Channel Name: " + channelName[i] + "<br> Signal Type: " + signalType[i] + "<br>
Physical Number: " + physicalNum[i] + "<br>Major Number: " + majorNum[i] + "<br> Minor Number: " + minorNum[i] + " ";
             }
             msgArea.innerHTML = htmlStr1 + "\n";
          }
```

```
        }
        showPropResult();

    }

    function setChannelHandler() {
        var channelList;
        var channelCount;
        var channelName;
        var signalType;
        var physicalNum;
        var majorNum;
        var minorNum;
        var logicalNum;
        var channelToSet = 2;

        channelList = broadcast.getAllChannelList("tv", "DIGITAL");
        jsLog.lgmethod("getAllChannelList()")
        if (channelList == undefined) {
            channelList = broadcast.getAllChannelList("tv", "ANALOG");
            jsLog.lgmethod("getAllChannelList()")
        }

        channelName = channelList[channelToSet].channelName;
        signalType = channelList[channelToSet].signalType;
        physicalNum = channelList[channelToSet].physicalNum;
        if (broadcast.isDvb()) {
            logicalNum = channelList[channelToSet].logicalNum; // For DVB,
            broadcast.setChannel(signalType, physicalNum, logicalNum);
            jsLog.lgmethod("setChannel()")
        }
        else {
            majorNum = channelList[channelToSet].majorNum; // For ATSC,
            minorNum = channelList[channelToSet].minorNum;
            broadcast.setChannel(signalType, physicalNum, majorNum, minorNum);
            jsLog.lgmethod("setChannel()")
        }
    }

    function programListHandler() {

        var today = new Date();
        var tYear = today.getYear() + 1900;
        var tMonth = modNumberToText(today.getMonth() + 1);
        var tDate = modNumberToText(today.getDate());
```

```
      var tHour = modNumberToText(today.getHours());
      var tMin = modNumberToText(today.getMinutes());
      var tSec = modNumberToText(today.getSeconds());
      var start_date = tYear + tMonth + tDate + tHour + tMin + tSec; //YYYYMMDDHHmmss
      var end_date = tYear + tMonth + tDate + "235959"; //YYYYMMDDHHmmss

      var programCount = broadcast.getProgramCount(start_date, end_date);
      jsLog.lgmethod('getProgramCount()');
      htmlStr1 += "<font style='color:#000000; font-weight:bold'>All Program Count: </font>" + programCount + "<br>";
      var programList = broadcast.getProgramList(start_date, end_date);
      jsLog.lgmethod('getProgramList()');
      var title = new Array();
      var startTime = new Array();
      var endTime = new Array();
      var description = new Array();

      if (programCount != 0) {
         htmlStr1 += "Program List: <br>";

         for (i = 0; i < programList.length; i++) {
            title[i] = programList[i].title;
            startTime[i] = programList[i].startTime;
            endTime[i] = programList[i].endTime;
            description[i] = programList[i].description;
            htmlStr1 += "[" + i + "] Title: " + title[i] + "<br>Start Time: " + startTime[i] + "<br>End Time: " + endTime[i] +
"<br>Description: " + description[i] + " <br>";
            msgArea.innerHTML = htmlStr1 + "\n";
         }
         showPropResult();
      }

   }

   function currentProgramInfoHandler() {
      var currentProgram = broadcast.getCurrentProgram();
      jsLog.lgmethod('getCurrentProgram()');
      htmlStr1 += "Title: " + currentProgram.title + "<br>Start Time: " + currentProgram.startTime + "<br>End Time: " +
currentProgram.endTime + "<br> Description: " + currentProgram.description + "<br> Event Id: " + currentProgram.eventId + " ";
      showPropResult();


   function nextProgramInfoHandler() {
      var nextProgram = broadcast.getNextProgram();
      jsLog.lgmethod('getNextProgram()');
      htmlStr1 += "Title: " + nextProgram.title + "<br>Start Time: " + nextProgram.startTime + "<br>End Time: " + nextProgram.endTime
```

```
+ "<br> description: " + nextProgram.description + "<br> Event Id: " + nextProgram.eventId + " ";
      showPropResult();

   }

   function setSizeHandler() {
      if ((broadcast.width == 240) && (broadcast.height == 180)) {
         broadcast.width = 320;
         broadcast.height = 240;
      }
      else {
         broadcast.width = 240;
         broadcast.height = 180;
      }
      jsLog.lgproperty('width:' + broadcast.width);
      jsLog.lgproperty('height:' + broadcast.height);
   }

   function checkNetCastVersion() {

      var nBrowserVersion = getBrowserVersion();
      if (nBrowserVersion == 4) // NetCast 2.0
      {
         document.getElementById('enable_numberkey').style.visibility = "hidden";
         document.getElementById('disable_numberkey').style.visibility = "hidden";
         document.getElementById('enable_channelKey').style.visibility = "hidden";
         document.getElementById('disableChannelKey').style.visibility = "hidden";
      }
      else if (nBrowserVersion >= 5) // NetCast 3.0 {
         addEventHandler(document.getElementById("set_channel"), "click", setChannelHandler);
         ddEventHandler(document.getElementById("program_list"), "click", programListHandler);
         addEventHandler(document.getElementById("current_programInfo"), "click", currentProgramInfoHandler);
         addEventHandler(document.getElementById('next_programInfo'), "click", nextProgramInfoHandler);
         addEventHandler(document.getElementById("all_channelList"), "click", allChannelListHandler);
         addEventHandler(document.getElementById("channelList"), "click", channelListHandler);
         addEventHandler(document.getElementById("set_size"), "click", setSizeHandler);

         broadcast.onchannelchange = processChannelChangeFunction;
         broadcast.onnosignal = processNoSignalFunction;
         broadcast.onchannelstatechange = processChStateChangeFunction;
         //broadcast.onepgupdate=processEpgupdate;

         setInnerTextById("method", "setChannel(),getChannelState()<br>getCurrentChannelName(),getCurrentChannelNumber()<br>
getAllChannelList(),getChannelList()<br>getChannelCount(),getCurrentProgram()<br>getNextProgram(),getProgramList()
<br>getProgramCount(),isChannelMapEmpty()
```

```
<br>isTunerInput(),isDvb()");
         setInnerTextById("property", "width<br>height");
         setInnerTextById("events", "onchannelchange<br>onnosignal<br>onchannelstatechange");
      }
   }

   function modNumberToText(num) {

      if (num < 10) num = "0" + num;

      return num;
   }

</script>
    </head>

    <body ondragstart='return false' onselectstart='return false'>
        <!-- title -->
        <div class='SuiteTitle'>LG Smart TV SDK | Web Open API Tutorial</div>
        <!-- navigation -->
        <div class='SuiteNavigation'>
            <div style="float:left;">File : broadcast/broadcast.html</div>
        </div>
        <div class='SuiteTitleLine'></div>
        <!-- test contents -->
        <div id='content_body' class='ContentArea'>
            <div class='ApiListTitleArea'>API List</div>
            <div class='ApiListArea'>
                <div class='MethodTitleArea'>Methods
                    <div class='MethodListArea' id="method">channelUp(), channelDown()
                        <br>
                    </div>
                </div>
                <div class='PropertyTitleArea'>Properties
                    <div class='PropertyListArea' id="property"></div>
                </div>
                <div class='EventTitleArea'>Events
                    <div class='EventListArea' id="events"></div>
                </div>
            </div>
            <div class='ViewTitleArea'>
                <div id='tabViewArea' class='SelectedViewArea' style='float:left;' onclick="showView();">View</div>
                <div id='tabCodeArea' class='UnselectedViewArea' style='float:right;'
                onclick="showCode();">Source</div>
            </div>
```

```
<div id='view'>
    <div class='ViewArea'>
        <table>
            <tr>
                <td style="width:350px; vertical-align:top">
                    <object id="broadcast" type="application/x-netcast-broadcast" width=240
                    height=180></object>
                </td>
                <td>
                    <table width="543" border="0">
                        <tr>
                            <td width="163">
                                <div id="channel_up" class="executeMiddleButton">Channel Up</div>
                            </td>
                            <td rowspan="11" width="380" style="vertical-align:top;">
                                <div id="displayArea" class="displayAreaStyle" style="overflow:auto;height:320px;"></div>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <div id="channel_down" class="executeMiddleButton">Channel Down</div>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <div id="set_channel" class="executeMiddleButton">Set Channel</div>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <div id="program_list" class="executeMiddleButton">Program List</div>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <div id="current_programInfo" class="executeMiddleButton">Current Program</div>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <div id="next_programInfo" class="executeMiddleButton">Next Program</div>
                            </td>
                        </tr>
                        <td>
                            <div id="all_channelList" class="executeMiddleButton">All Channel List</div>
```

```
                                    </td>
                        </tr>
                        <tr>
                            <td>
                                <div id="channelList" class="executeMiddleButton">Channel List</div>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <div id="set_size" class="executeMiddleButton">Set Size</div>
                            </td>
                        </tr>
                        <tr>
                            <td></td>
                        </tr>
                        <tr height="50px">
                            <td colspan="2" align=left>
                                <div id="APIdescription"></div>
                            </td>
                        </tr>
                        </table>
                        </td>
                        </tr>
                    </table>
                </div>
            </div>
            <div style="visibility: hidden" id='codeview'>
                <textarea class="SourceCodeArea" value="" id='sourcecode'></textarea>
            </div>
        </div>
        <!-- button and copyright -->
        <div class='SuiteButtonArea'>
            <!-- button -->
            <div id='btn_back' class='buttonDescription'>BACK</div>
            <!-- exit key description -->
            <div id='btn_exit' class='buttonDescription'>EXIT</div>
            <!--<div id='btn_yellow' class='buttonDescription yellowColor'>NEXT PAGE</div>-->
            <!-- copyright -->
            <div class='copyright'>Copyright LG Electronics</div>
        </div>
    </body>

</html>
```