Creating a Video Application With HAS (HTTP Adaptive Streaming)

Published 2014-10-28 | (Compatible with SDK 2.5,3.5,4.5,5.0,5.1 and 2011,2012,2013,2014 models)

> Tutorial describes the use of the HAS (HTTP Adaptive Streaming) features of the Samsung Applications service and demonstrates how to create an application featuring HAS content playback

Contents

This tutorial demonstrates how you can create an application featuring HAS content playback. The application reads content list information from a video list file in XML format, and the user can play, stop, pause, resume, and do forwards and backwards skip of the content. The application has two video display modes - window mode and full screen mode.

The application has a similar GUI and functionality as the Creating a Video Application tutorial. The most significant differences between this tutorial and the Video tutorial are the video list URL and the bitrates graph. The playback of HAS contents is supported by the AVPlay class. This means that all play control of HAS contents is same as for normal content.

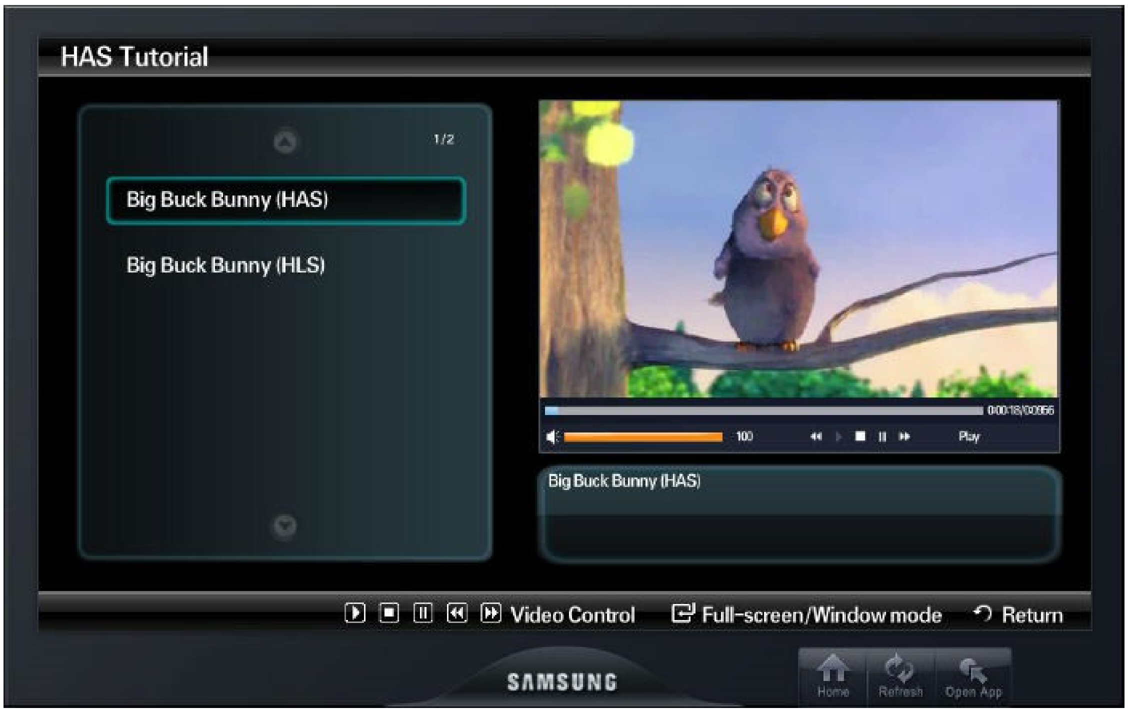

Figure. HAS Content Play - Window mode

Playing the HAS content on full screen mode, displays the bitrates bar graph as shown in the figure below. The blue-colored bitrates bar in the figure below indicates bitrates of the current selected stream and is updated every second. When the network bandwidth changes, the bitrate bar changes accordingly. Note that the bitrates bar changing time does not match with the video quality changes on the screen. The time gap varies depending on the size of buffered stream in the HAS module.

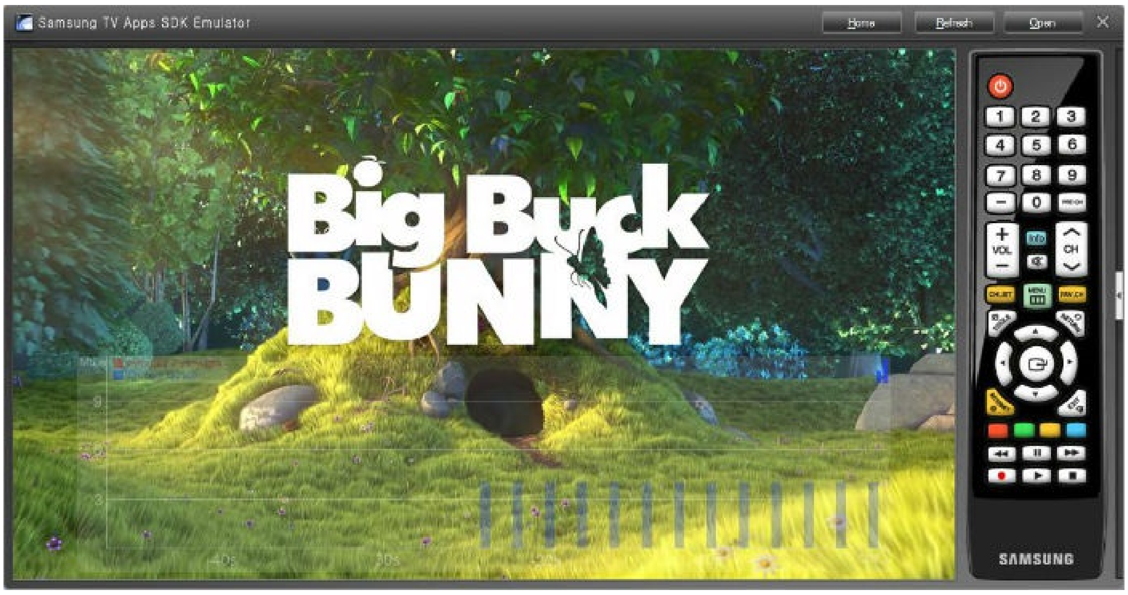

Figure. HAS Content Play - Full-Screen mode

Prerequisites

To create applications that run on a TV screen, you need:

Samsung TV connected to the Internet

SDK or a text editor for creating HTML, JavaScript and CSS files (using Samsung Smart TV SDK is recommended)

Development environment

Use Samsung Smart TV SDK to create the application. You can also use the emulator provided with the SDK to debug and test the application before uploading it in your TV. Later, you can run the application on a TV; see Testing Your Application on a TV. Note that applications may perform better on the TV than on the emulator. You can find general instructions for creating applications in Implementing Your Application Code.

Source Files

With this tutorial application, sample HAS compatible and HLS compatible contents can be played. To play HAS or HLS (HTTP Live Streaming) contents on the HTTP server, prepare HAS or HLS compatible contents and metadata files conforming to the standards.

> Note
>
> The files needed for the sample application are here: **Tutorial HAS Source**.

The directory structure for the tutorial application:

Directory	Description
Common/API	Contains the common modules provided by the Application Manager.
CSS	Contains the CSS files.
Images	Contains the image files.
Javascript	Contains the JavaScript files.
XML	Contains the videoList.xml file.

Design

The HAS application is composed of the following parts, which are displayed in the figure below:

TV

controlling the parts of the TV

Network

accessing data from the network

Graphics

displaying information graphically in the browser

Control

managing actions in response to user input

Bitrate

displaying the content bitrate graph

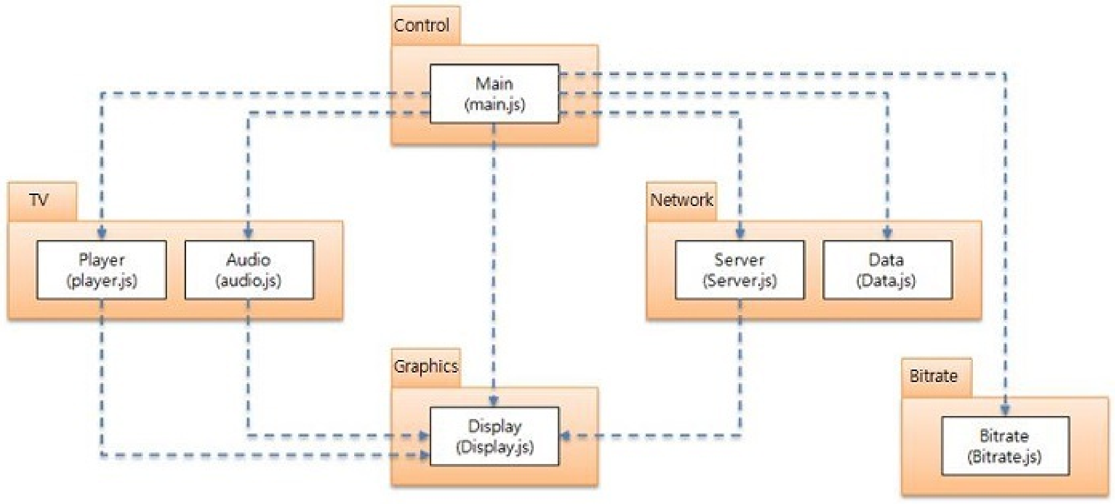

Figure. Design Diagram

Class description

The classes that form part of the HAS application are as follows:

Class	Description
Main	Responsible for key handling and coordination of all the application parts.
Player	Controls the audio and video playback from the content server using an API.

Class	Description
Audio	Controls the audio volume level using an API.
Server	Handles the retrieval of RSS feed from the data server using AJAX.
Data	Handles the storage of video data within the application.
Display	Responsible for displaying the graphics and text information using dynamic HTML.
Bitrate	Responsible for displaying the current content bitrates bar graph in Full screen mode.

Video Application With HAS (Http Adaptive Streaming)

This section briefly describes the initial configuration of the application. For details on basic application creation, see Application Development Process.

Creating the Basic Application

To create the basic application,

1. Start the SDK for Samsung TV Apps.
2. Create a new application using the following config.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<widget>
  <previewjs>PreviewHASTutorial</previewjs>
  <type>user</type>
  <cpname></cpname>
  <cplogo></cplogo>
  <cpauthjs></cpauthjs>
  <ver></ver>
  <mgrver></mgrver>
  <fullwidget>y</fullwidget>
  <movie>y</movie>
  <srcctl>n</srcctl>
  <ticker>n</ticker>
  <childlock>n</childlock>
  <audiomute>n</audiomute>
  <videomute>n</videomute>
  <dcont>y</dcont>
  <widgetname>HAS Tutorial</widgetname>
  <description></description>
  <width>960</width>
  <height>540</height>
  <author>
    <name>Samsung Electronics Co. Ltd.</name>
    <email></email>
    <link>http://www.sec.com</link>
    <organization>Samsung Electronics Co. Ltd.</organization>
  </author>
</widget>
```

   The following settings are used:

   <fullwidget>y</fullwidget>

   Makes the application run in full screen mode. This affects what keys are registered by default.

   <type>user</type>

   Enables the user application feature for testing on a real TV set. This tag has no effect on the emulator.

   <movie>y</movie>

Optimises the application behaviour for movie playback.

HAS Content

The HAS content items for this application are listed in the videoList.xml file. The code is given below.

```
<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <!—The url below is a sample URL which is compatible with the OIPF HAS Rel2 specification.-->
    <item>
      <title>Big Buck Bunny (HAS)</title>
      <link>http://drq08ysq9m8gx.cloudfront.net/TestStream/HAS_BigBuckTS.xml|COMPONENT=HAS</link>
      <description>Big Buck Bunny (HAS)</description>
    </item>
    <!-- The url below url is a sample URL which is compatible with the APPLE HLS (Http Live Streaming) specificatio
n.-->
   <item>
     <title>Big Buck Bunny (HLS)</title>
      <link>http://drq08ysq9m8gx.cloudfront.net/TestStream/HLS_BigBuckTS.m3u8|COMPONENT=HLS</link>
      <description>Big Buck Bunny (HLS)</description>
   </item>
</channel>
</rss>
```

The videoList.xml file includes 2 content items - one for HAS type content and another for HLS type content. The contents items described in this file are displayed on the video selection window. Other test contents can be added to this list. The HAS contents URL must have Samsung-specific URL parameters beginning with | character. The COMPONENT parameter indicates the standard type with which the content is compatible. Current Samsung TV models support two adaptive standards - OIPF HAS and Apple HLS. |COMPONENT=HAS indicates OIPF HAS compatible contents and |COMPONENT=HLS indicates Apple HLS compatible contents. For more details about other Samsung-specific URL parameters, see URL Parameters for HTTP Adaptive Streaming Content. The test contents are explained in the table below.

Title	Description
Big Buck Bunny (HAS)	OPIF HAS compatible adaptive streaming content. Have three bitrates (0.5Mbps, 1Mbps, 2 Mbps) stream.
Big Buck Bunny (HLS)	Apple HLS compatible adaptive streaming content. Have three bitrates (0.5Mbps, 1Mbps, 2 Mbps) stream.

HLS type : Apple HTTP Live streaming (draft-pantos-http-live-streaming-04.txt) The GUI, Javascript, CSS and other components for playing a video, video control and feedback, volume control, video selection, and screen modes are same as those described in Creating a Video Application. When screen mode changes from window mode to full screen mode, the content bitrates bars are overlaid on top of the movie.

Content Bitrates Graph Display

When the network bandwidth changes, the HAS engine adaptively selects the most suitable bitrates stream from the available alternatives. The Bitrate.js file displays the content bitrates changes on top of the video screen in full screen mode while playing the contents. The Bitrate class is an utility class for drawing a content bitrate graph on the screen. Its main task is to periodically get the content bitrates value and draw the content bitrate bar on the screen. The usage sequence for the Bitrate class is given below.

1. After starting the content playback, call the init() and startMonitor functions.

   Main.js ::keyDown()

```
case tvKey.KEY_PLAY:
    alert("PLAY");
    this.handlePlayKey();
    Bitrate.init();
    Player.playVideo();
    Bitrate.startMonitor();
```