Media Control

Programming Guide

Version 1.0

Table of Contents

1. Overview

Media Control allows you to easily develop applications that share videos, music, and photos between network devices.

You can use Media Control to:

- play media files stored on a network device or web server on another device
- search for and browse media files on a content provider
- control the media files using playback control functions such as pause, resume, seek and stop for music and video files

1.1. Architecture

The following figure shows the Media Control architecture.

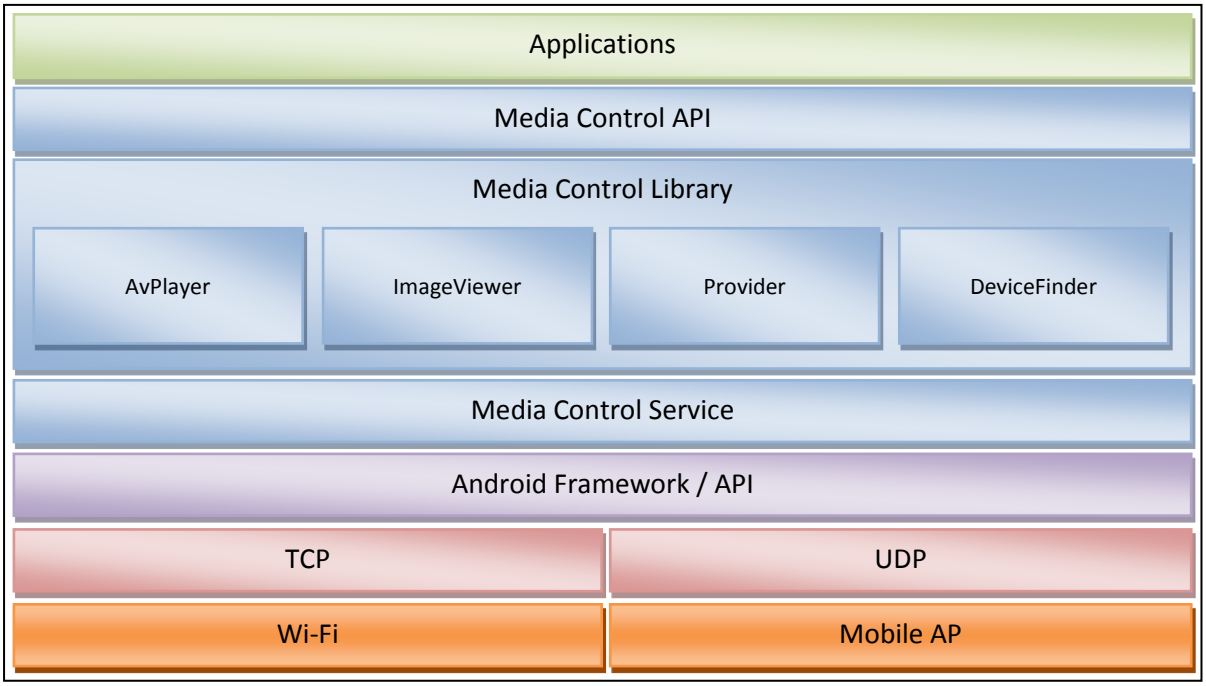

Figure 1: Media Control architecture

- **Applications:** One or more applications that use Media Control.
- **Media Control API**: Application Programming Interface that gives access to Media Control components.
- **Media Control Library**: Set of components that implement devices (AvPlayer, ImageViewer, Provider) and device finder.
    - **AvPlayer:** Software component to stream audio or video and control the audio player.
    - **ImageViewer:** Software component to show images and control the image viewer.

- **Provider:** Software component to access the content provider.
- **DeviceFinder:** Software component to discover devices.

- **Media Control Service**: Media Control Android service.

1.2. Class Diagram

The following figure shows the Media Control classes and interfaces that you can use in your application.

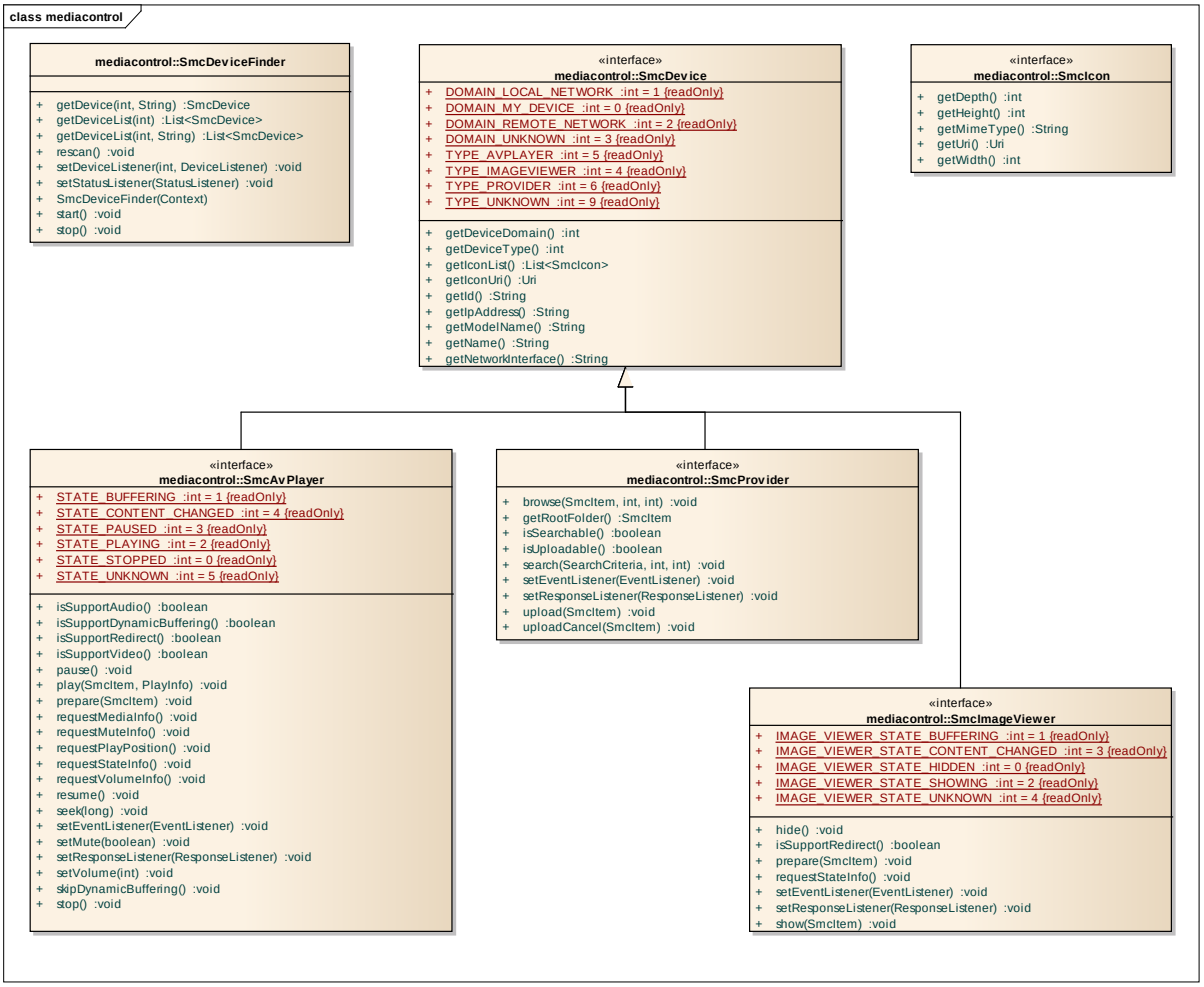

Figure 2: Media Control classes and interfaces

The Media Control classes and interfaces include:

- **SmcDeviceFinder:** Discovers Media Control devices in the local network.
- **SmcDevice:** Provides Media Control device information and type casting for device types such as SmcAvPlayer.
- **SmcItem:** Manages content metadata.

- **SmcAvPlayer:** Streams audio/video content to a remote audio/video player.
- **SmcImageViewer:** Shows image content on a remote image viewer.
- **SmcProvider:** Provides access to remote content providers.

1.3. Supported Platforms

Android 4.1 (Jelly Bean API 16) or above support Media Control.

1.4. Supported Features

Media Control supports all the functionality defined by the DLNA standard, including:

- displaying images
- playing video and music files
- browsing and playing DMS (Digital Media Servers) contents

1.5. Components

- Components
  - mediacontrol-v1.0.0.jar
- Imported packages:
  - com.samsung.android.sdk.mediacontrol

1.6. Installing the Package for Eclipse

To install Media Control for Eclipse:

1. Add the mediacontrol-v1.0.0.jar file to the libs folder in Eclipse.

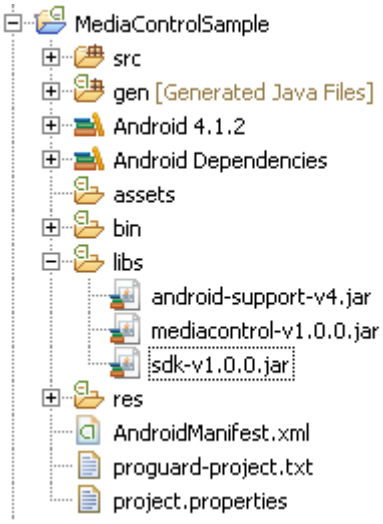

Figure 3: libs folder in Eclipse

2. Add the following permissions to your Android manifest file:

```
<uses-permission android:name="com.sec.android.permission.PERSONAL_MEDIA"/>
<uses-permission android:name="com.sec.android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
```

2. Hello Media Control

Hello Media Control is a simple program that:

- Connects to the Media Control Service
- Starts the device finder
- Gets a device by id (ImageViewer in the sample)
- Sets an image to SmcItem and shows the image item in an image viewer.

```
public class MainActivity extends Activity {
// Current list of devices(Image Viewer)
private final List<SmcDevice> mDevices = new ArrayList<SmcDevice>();
// lib
private Smc mSmcLib;
// current device finder
private SmcDeviceFinder mDeviceFinder;
// current Image viewer
private SmcImageViewer mImageViewer;
// Image viewer listener
ImageViewerListener mImageViewerListener;
// Path to image displayed on ImageViewer
final String filePath = "/storage/emulated/0/Download/test.jpg";
// Mimetype of file
final String mimetype = "image/jpeg";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Create an instance of Smc instance
    mSmcLib = new Smc();
    try {
        // Connecting to the Media Control Service
        mSmcLib.initialize(getBaseContext());
    } catch (SsdkUnsupportedException e) {
        e.printStackTrace(); // TODO Handle exceptions.
    }
    // create device finder,
    mDeviceFinder = new SmcDeviceFinder(this);
    mDeviceFinder.setStatusListener(mDeviceFinderStateListener);
    mDeviceFinder.start();
}

@Override
protected void onDestroy() {
    super.onDestroy();
    hide();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
```

```
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    // Create status listenerdevice finder started - scan the device list
    SmcDeviceFinder.StatusListener mDeviceFinderStateListener = new
SmcDeviceFinder.StatusListener() {
        @Override
        public void onStarted(SmcDeviceFinder deviceFinder, int error) {
            if (error == Smc.SUCCESS) {
                mDeviceFinder = deviceFinder;
                mDeviceFinder.setDeviceListener(SmcDevice.TYPE_IMAGEVIEWER,
mDeviceListener);
                mDeviceFinder.rescan();
            }
        }

        @Override
        public void onStopped(SmcDeviceFinder deviceFinder) {
            if (mDeviceFinder == deviceFinder) {
                mDeviceFinder.setDeviceListener(SmcDevice.TYPE_IMAGEVIEWER,
null);
                mDeviceFinder.setStatusListener(null);
                mDeviceFinder = null;
            }
        }
    };

    // get Create device listenerimageviewer list
    DeviceListener mDeviceListener = new DeviceListener() {
        @Override
        public void onDeviceAdded(SmcDeviceFinder deviceFinder, SmcDevice
smsDevice) {
            mDevices = mDeviceFinder.getDeviceList(SmcDevice.TYPE_IMAGEVIEWER);
            show();
        }

        @Override
        public void onDeviceRemoved(SmcDeviceFinder deviceFinder, SmcDevice
SmsDevice, int error) {
            mDevices = mDeviceFinder.getDeviceList(SmcDevice.TYPE_IMAGEVIEWER);
        }
    };

    private void registerImageViewerListener() {
        if (mImageViewer != null && mImageViewerListener == null) {
            mImageViewerListener = new ImageViewerListener();
            mImageViewer.setEventListener(mImageViewerListener);
        }
    }

    private void unregisterImageViewerListener() {
        if (mImageViewer != null) {
            mImageViewer.setEventListener(null);
            mImageViewerListener = null;
        }
    }

    // Listener responsible for handling ImageViewer events
```

```
    private class ImageViewerListener implements SmcImageViewer.EventListener {
        @Override
        public void onDeviceChanged(SmcImageViewer device, int imageViewerState,
int error) {
            switch (imageViewerState) {
            case SmcImageViewer.IMAGE_VIEWER_STATE_BUFFERING:
                break;
            case SmcImageViewer.IMAGE_VIEWER_STATE_SHOWING:
                break;
            case SmcImageViewer.IMAGE_VIEWER_STATE_CONTENT_CHANGED:
                break;
            }
        }
    };

    private void show() {
        if (mDeviceFinder != null) {
            if (mDevices.size() > 0) {
                mImageViewer = (SmcImageViewer)mDevices.get(0);
            }
            if (mImageViewer != null) {
                registerImageViewerListener();
                SmcItem item = new SmcItem(new SmcItem.LocalContent(filePath,
mimetype));
                mImageViewer.show(item);
            }
        }
    }

    private void hide() {
        if (mImageViewer != null) {
            mImageViewer.hide();
        }
        unregisterImageViewerListener();
        mDeviceFinder.stop();
    }
}
```

3. Initializing the Media Control Package

You need to initialize a Smc before you can use it. Samsung Mobile SDK provides a base class with an initialize() method for each package.

The Smc can run only on Samsung Smart Devices. Some Samsung Smart Device models do not support some *of the* packages.

You can use a `initialize()` method to initialize it and also to check if the device supports the Smc. If the device does not support the Smc, the method throws an SsdkUnsupportedException exception. You should handle this exception. If an SsdkUnsupportedException exception is thrown, you can check the exception type with **`SsdkUnsupportedException.getType()`**. If the device is not a Samsung device, the exception type is **SsdkUnsupportedException.VENDOR_NOT_SUPPORTED**. If the device is a Samsung model that does not support the Smc, the exception type is **SsdkUnsupportedException.DEVICE_NOT_SUPPORTED**.

The Smc class provides the following methods:

- `initialize()` initializes Media Control and connects to the Media Control Service.
- `getVersionCode()` returns the Media Control version number as an integer.
- `getVersionName()` returns the Media Control version name as a string.
- `isFeatureEnabled(int type)` checks if a Media Control package feature is available on the device.

```
// Create an Smc instance
mSmcLib = new Smc();
try {
    // Connect to the Media Control Service
    mSmcLib.initialize(getBaseContext());
} catch (SsdkUnsupportedException e) {
    e.printStackTrace();  //TODO Handle exceptions.
}
int versionCode = mSmcLib.getVersionCode();
String versionName = mSmcLib.getVersionName();
```

3.1. Using the Initialization Method

The initialization method provides the following functionality:

- Initializes the package.
- Checks if the device is a Samsung Device.
- Checks if the device supports the package.
- Checks if the package libraries are installed on the device.

```
void initialize(Context context) throws SsdkUnsupportedException
```

If the method fails to initialize the Smc, it throws an **SsdkUnsupportedException** exception. If an exception is thrown, you should check the exception message type.

3.2. Ssdk UnsupportedException Messages

SsdkUnsupportedException exceptions are generated if the initialization method fails to initialize the Smc. The class defines the following exception types:

- **VENDOR_NOT_SUPPORTED:** The device is not a Samsung device.
- **DEVICE_NOT_SUPPORTED:** The device is a Samsung device that does not support the package.

3.3. Checking Feature Availability

You can use the isFeatureEnabled method to check if the device supports the feature. Define the type for each feature in the base class of the Smc and pass the type as a parameter in this method to check if the feature can run on the device. The method returns a Boolean value that indicates if the feature is available. Smc defines the following feature types:

- **FEATURE_AVPLAYER:** The device is support the AV player.
- **FEATURE_IMANGEVIEWER:** The device is support the Image viewer.
- **FEATURE_PROVIDER:** The device is support the contents provider.

```
boolean isFeatureEnabled(int type);
```

4. Using the Media Control Package

This section describes how to use the Media Control package to develop your application.

4.1. Discovering Media Control Devices

To discover devices that are part of the Media Control Service:

1. Create an SmcDeviceFinder instance to retrieve SmcDevice instances from the Media Control Service. The SmcDeviceFinder class offers the following methods.

Method	Description
`getDevice(int deviceType, String id)`	Return the device with the specified device ID and device type.
`getDeviceList(int deviceType)`	Return the list of discovered devices with the specified device type.
`getDeviceList(int deviceType, String networkInterface)`	Return the list of discovered devices with the specified device type within the specified network interface controller.

2. Create a `SmcDeviceFinder.DeviceListener` listener and register it with your SmcDeviceFinder instance to receive device availability events. The onDeviceAdded() or onDeviceRemoved() methods are called when a device is added or removed in the Media Control network.

You can use the rescan() method to refresh the existing device list. The Media Control package sends a broadcast message in the network and the result is reported through the `SmcDeviceFinder.DeviceListener` listener.

```
    SmcDeviceFinder.DeviceListener listener = new
SmcDeviceFinder.DeviceListener() {
        @Override
        public void onDeviceAdded(SmcDeviceFinder deviceFinder, SmcDevice
smcDevice) {
            // Device newly found in framework service.
            // Update device list information in the application.
        }
        @Override
        public void onDeviceRemoved(SmcDeviceFinder deviceFinder, SmcDevice
smcDevice, int error) {
            // Device disappeared in the network.
            // Update device list information in the application.
        }
    };

    //Register the DeviceListener listener
    mDeviceFinder.setDeviceListener(SmcDevice.TYPE_AVPLAYER, listener);
    //Get device list from the Media Control package
```

```
mDevices = mDeviceFinder.getDeviceList(SmcDevice.TYPE_AVPLAYER);
//Refresh the device list
mDeviceFinder.rescan();
```

The Media Control package provides several Media Control device types. After getting an SmcDevice instance, you can type cast it by device type.

Class	Device Type Enum	Description
SmcAvPlayer	TYPE_AVPLAYER	Remote Audio and Video Player Device
SmcImageViewer	TYPE_IMAGEVIEWER	Remote Image Viewer Device
SmcProvider	TYPE_PROVIDER	Remote Content Provider Device

4.2. Getting Device Information

You can get information on each device with the following SmcDevice methods.

Method	Description	Return Value
`getDeviceDomain()`	Return the device network domain.	int Example: DEVICE_DOMAIN_LOCAL_NETWORK
`getDeviceType()`	Return the device type.	int Example: DEVICE_TYPE_IMAGEVIEWER
`getIconUri()`	Return the representative icon URI of the device.	Example: “http://128.202.198.47:17676/DeviceIcon”
`getIconList()`	Return the list of all icons of the device.	A list of SmcIcon
`getId()`	Return the device unique ID.	string Example: “uuid:08f0d180-0002-1000-8823-00245491c0ab”
`getModelName()`	Return the device model name.	Example: “Samsung Smart TV”
`getName()`	Return the user specified device name.	Example: “[TV]Smart TV”
`getNetworkInterface()`	Return the network interface to which the device is connected.	Network adapter name string
`getIpAddress()`	Return the device IP address.	Example: “111.22.33.4”

For information on how to download a device icon from a Media Control device, see the sample code at:

http://developer.android.com/resources/samples/XmlAdapters/src/com/example/android/xmladapters/ImageDownloader.html

4.3. Streaming Media Files to a Media Control Device

The Media Control package allows media files, which can be stored on a local drive, on a Media Control Service device, or on the Internet, to be played on a remote player device.

To play a media file on a remote player device:

3. Create an SmcItem instance.

The methods for creating an SmcItem instance depend on the location of the content. If the content is stored on a local device, then provide the file path and MIME type of the content. If the content is stored remotely, then provide the URI and MIME type.

The SmcItem class provides two constructors:

- For creating local items, use SmcItem.LocalContent.
- For creating remote content, use SmcItem.WebContent.

The first argument (file path) for the LocalContent constructor is a URI with a content scheme or a full file path. The second argument is the MIME type. You can get the MIME type from the Android MediaStore, but this step is omitted in this section.

```
    // Content in media db
    String contentLocation ="content://media/external/images/media";
    Context context;

    // Retrieve the path and the MIME type from media db
    Uri uri = Uri.parse(contentLocation);
    Cursor cursor = context.getContentResolver().query( uri, null, null, null,
null );
    while(cursor.moveToNext())
    {
        String filepath =
          ((Object)
cursor).getData( cursor.getColumnIndex( MediaStore.MediaColumns.DATA ));
        String mimeType =

cursor.getString( cursor.getColumnIndex( MediaStore.MediaColumns.MIME_TYPE ) );
    }

    // Content which is not registered in media db
```

```
String filePath = "/mnt/sdcard/download/_image/1-image.jpg";
String mimeType = "image/jpeg";

// Build the item
SmcItem.LocalContent content = new SmcItem.LocalContent(filePath, mimeType);
    SmcItem item = new SmcItem(content);
```

For more information on MediaStore, see http://developer.android.com/reference/android/provider/MediaStore.html

If the content is stored on the Internet and accessible with Uri, the first argument is the URI of the web content. For the MIME type, see the content type of the Web content.

```
contentLocation = "http://...";       // a content URI on the Internet
uri = Uri.parse(contentLocation);
mimeType = "image/*";

SmcItem.WebContent webContent = new SmcItem.WebContent(uri, mimeType);
SmcItem webContents = new SmcItem(webContent);
```

You can set the subtitle file for a video item. If you do not set the subtitle explicitly, the Media Control package creates the subtitle with same file name automatically.

```
contentLocation = "http://..."; // a content URI on the Internet
uri = Uri.parse(contentLocation);
mimeType = "image/*";
subTitlePath = "...";

SmcItem.WebContent webContent = new SmcItem.WebContent(uri, mimeType)
                                    .setSubTitle(subTitlePath);
SmcItem webContents = new SmcItem(webContent);
mImageViewer.show(webContents);
```

Media Control provides the following content delivery modes:

- **Relay delivery mode:** If you set the delivery mode to relay mode, you can stream the content to the player via the device.

  DELIVERY_MODE_RELAY // set to relay delivery mode

- **Redirect delivery mode:** If you set the delivery mode to redirect mode, the player accesses the web server directly.

  Call the `SmcAvPlayer.isSupportRedirect()` method to check if the player supports redirect mode.

  DELIVERY_MODE_REDIRECT // set to redirect delivery mode

Call the setter methods to add metadata. If the player supports the metadata, it displays the information.

```
contentLocation = "http://..."; // a content URI on the Internet
uri = Uri.parse(contentLocation);
mimeType = "image/*";
albumTitle = "TestTitle";
artist = "TestArtist" ;

SmcItem.WebContent webContent = new SmcItem.WebContent(uri, mimeType)
                .setDeliveryMode(WebContent.DELIVERY_MODE_RELAY)
                .setAlbumTitle(albumTitle)
                .setArtist(artist);
SmcItem webContents = new SmcItem(webContent);
mImageViewer.show(webContents);
```

4. Select the target remote device to play the item.
5. To get a list of target devices, use the SmcDeviceFinder class, specifying the type of device for your application. Use SmcDeviceType.TYPE_IMAGEVIEWER for SmcImageViewer devices and SmcDeviceType.TYPE_AVPLAYER for SmcAvPlayer devices.

```
java.util.List<SmcDevice> avPlayerList =
    mDeviceFinder.getDeviceList(SmcDevice.TYPE_AVPLAYER);
java.util.List<SmcDevice> imageViewerList =
        mDeviceFinder.getDeviceList(SmcDevice.TYPE_IMAGEVIEWER);
```

6. To specify the starting position of the audio/video playback, use the PlayInfo object. You can create an SmcAvPlayer.PlayInfo instance as shown below.

```
SmcAvPlayer.PlayInfo playInfo = new SmcAvPlayer.PlayInfo(100);
player.play(webContents, playInfo);
```

SmcAvPlayer provides the `play()`, `stop()`, and `seek()` methods for controlling content playback. SmcImageViewer provides the `show()` and `stop()` methods.

7. Register an event listener to receive player events. Because communications between mobile and external devices may take a long time, the results are returned asynchronously. Each device provides interfaces for response listeners.

```
SmcImageViewer imageViewer = (SmcImageViewer) mDevices.get(0);
imageViewer.setResponseListener( new SmcImageViewer.ResponseListener()
{
    @Override
    public void onShow(SmcImageViewer imageViewer, SmcItem item, int error) {
        Log.d( TAG, "Show Response Received:" + error );
    }
    @Override
    public void onHide(SmcImageViewer imageViewer, int error) {
        Log.d( TAG, "Hide Response Received:" + error );
    }
```

```
        @Override
        public void onRequestStateInfo(SmcImageViewer imageViewer, int state, int
error) {
                Log.d( TAG, "Statecd Response Received:" + error );
            }
            } );
```

8. After registering the event listener, send the control commands to the device (SmcImageViewer or SmcAvPlayer). To play the content, call

- `void play (SmcItem item, SmcAvPlayer.PlayInfo playInfo)` for an SmcAvPlayer device.
- `void show (SmcItem item, SmcAvPlayer.PlayInfo playInfo)` for an SmcImageViewer device.

9. Media Control provides methods to check the state of SmcImageViewer and SmcAvPlayer. When you play music on a TV for example, the TV play state may change to stop automatically at the end of the music playback even if you do not request it. To handle this, register an event listener for the device to receive device status change notifications.

```
    player.setEventListener(new SmcAvPlayer.EventListener()
    {
        @Override
        public void onDeviceChanged(SmcAvPlayer device,
        int state, int error )
        {
            Log.d(TAG, "AV Player Event received: " + state);
        }
        } );
```

The following table lists the SmcAvPlayer states and the corresponding actions available.

State	Description	Available Actions
STATE_STOPPED	Content has stopped playing	`stop(), play(), seek()`
STATE_BUFFERING	Content is buffering	`stop(), seek()`
STATE_PLAYING	Content is playing	`stop(), pause()`
STATE_PAUSED	Content paused	`stop(), play()`
STATE_UNKNOWN	Unknown state	`stop()`
STATE_CONTENT_CHANGED	Content changed by another device	`stop()`

- STATE_UNKNOWN: Indicates that there is a network connection problem and your application is not able to determine its state.
- STATE_CONTENT_CHANGED: Indicates that the content of the device has been changed by another device. If your application receives this event, initialize the playing status UI.

The following diagram shows the state flow for SmcAvPlayer.

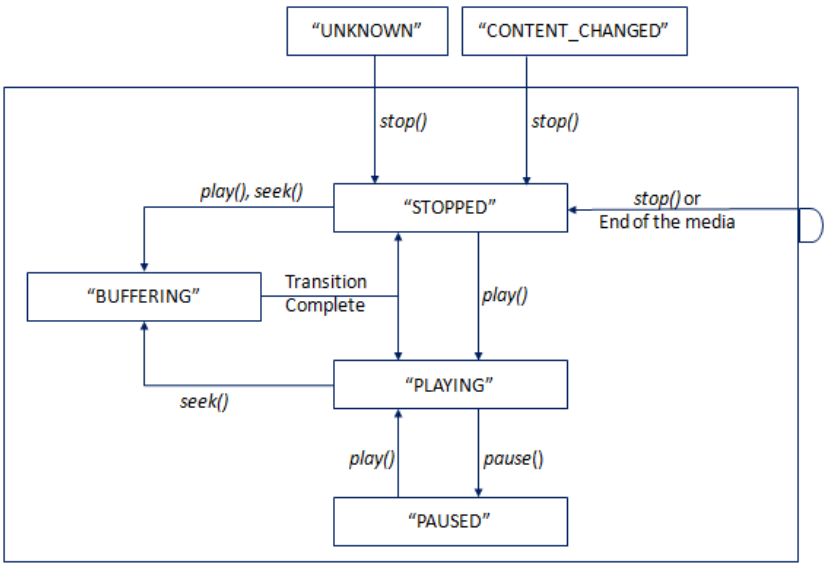

Figure 4: State flow for SmcAvPlayer

The following table lists the SmcImageViewer states and the corresponding actions available.

State	Description	Available Actions
IMAGE_VIEWER_STATE_UNKNOWN	State is unknown	`stop()`
IMAGE_VIEWER_STATE_STOPPED	Stopped	`stop(), show()`
IMAGE_VIEWER_STATE_BUFFERING	Image buffering	`stop()`
IMAGE_VIEWER_STATE_SHOWING	Image is displayed	`stop(), show()`
IMAGE_VIEWER_STATE_CONTENT_CHANGED	Content changed by another device	`stop()`

- IMAGE_VIEWER_STATE_UNKNOWN: Indicates that there is a network connection problem and your application is not able to determine its state.
- IMAGE_VIEWER_STATE_CONTENT_CHANGED: Indicates that the content of the device has been changed by another device. If your application receives this event, initialize the playing status UI.

The following diagram shows the state flow for the SmcImageViewer.

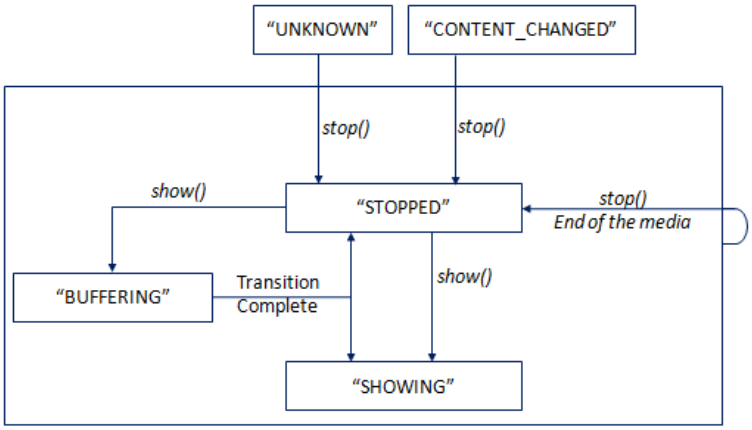

Figure 5: State flow of SmcImageViewer

The status of the device can be changed:

- when an application executes a command
- by the logic of the device itself
- by another controller

4.4. Browsing and Sharing Media Contents from a Media Control Service Device

To get a content list from a media content provider:

10. Call the browse and search asynchronous methods of the SmcProvider class.

- The browse method returns a list of content items according to the directory structure in the Media Control Service device.
- The search method returns contents matching the specified search conditions.

The following sample code shows how to use the browse() method.

```
    List<SmcDevice> providerList =
mDeviceFinder.getDeviceList(SmcDevice.TYPE_PROVIDER);
    SmcProvider selectedProvider = (SmcProvider) providerList.get(0);

    // enter root folder as input for the first browse on a provider
        selectedProvider.browse(selectedProvider.getRootFolder(), 0, 10);
```

The browse method requires the folder to browse, the starting item index to browse and the number of items as input parameters. The Media Control package returns the requested number of items from the content list in the BrowseResponseListener listener. The number of items to browse affects the browse response speed. Set it accordingly.

11. If you are using the browse method, register a listener to receive browse events.

```
    SmcProvider.ResponseListener mProviderResponseListener = new
SmcProvider.ResponseListener()
```

```
    {
        @Override
        public void onBrowse(SmcProvider device,
        List<SmcItem> smcItems, int requestedStartIndex,
            int requestCount, SmcItem requestedFolderSmcItem,
            boolean endOfItems, int error)
        {
        }
        @Override
        public void onSearch(SmcProvider smcProvider, List<SmcItem> smcItems,
            int requestedStartIndex, int requestedCount,
            SmcProvider.SearchCriteria searchCriteria, boolean endOfItems, int
error) {
        }

        @Override
        public void onUpload(SmcProvider smcProvider, SmcItem smcItem, int error)
{
        }
        @Override
            public void onUploadCancel(SmcProvider smcProvider, SmcItem smcItem,
int error) {
            }
        };
            selectedProvider.setResponseListener(mProviderResponseListener);
```

The following table lists the item types.

Item Type Enum	Description
MEDIA_TYPE_ITEM_FOLDER	Folder item, which may contain any other Item
MEDIA_TYPE_ITEM_AUDIO	Audio item
MEDIA_TYPE_ITEM_IMAGE	Image item
MEDIA_TYPE_ITEM_VIDEO	Video item

The following sample code shows how to use the search() method.

```
    List<SmcDevice> providerList =
mDeviceFinder.getDeviceList(SmcDevice.TYPE_PROVIDER);
SmcProvider selectedProvider = (SmcProvider)providerList.get(0);

    SearchCriteria searchCriteria = new SearchCriteria("love");
    int startIndex = 0, requestCount = 0;
        selectedProvider.search(searchCriteria,startIndex, requestCount);
```

The search() method requires a search criteria, the starting item index to search, and the number of search items as input parameters. The Media Control package returns the requested number of items from the content list in the ResponseListener listener. The list starts with the specified starting index.

You can create an SearchCriteria instance with keyword and item type. If you do not add a media type using the addItemType() method, all the media content related to the keyword is returned. If you do not set the keyword, all the content with the specified media type is returned.

12. If you are using the search method, register a listener to receive the search results:

```
    SmcProvider.ResponseListener mSearchResponseListener = new
SmcProvider.ResponseListener ()
    {
        @Override
        public void onSearch(SmcProvider smcProvider, List<SmcItem> smcItems,
            int requestedStartIndex, int requestedCount,
            SmcProvider.SearchCriteria searchCriteria, boolean endOfItems, int
error)
        {
        }

        @Override
        public void onBrowse(SmcProvider smcProvider, List<SmcItem> smcItems,
            int requestedStartIndex, int requestedCount,
            SmcItem requestedFolder, boolean endOfItems, int error)
        {
        }
        @Override
        public void onUpload(SmcProvider smcProvider, SmcItem smcItem, int error)
{
        }
        @Override
        public void onUploadCancel(SmcProvider smcProvider, SmcItem smcItem, int
error) {
        }
    };
        selectedProvider.setResponseListener(mSearchResponseListener);
```

You can upload a media file to a Media Control device by calling the upload() methods.

```
List< SmcDevice > providerList =
    mDeviceFinder.getDeviceList(SmcDevice.TYPE_PROVIDER);
SmcProvider provider = (SmcProvider)providerList.get(0);
SmcItem rootItem = provider.getRootFolder();
provider.upload(rootItem);
```

Copyright

Though every care has been taken to ensure the accuracy of this document, Samsung Electronics Co., Ltd. cannot accept responsibility for any errors or omissions or for any loss occurred to any person, whether legal or natural, from acting, or refraining from action, as a result of the information contained herein. Information in this document is subject to change at any time without obligation to notify any person of such changes.

Samsung Electronics Co. Ltd. may have patents or patent pending applications, trademarks copyrights or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give the recipient or reader any license to these patents, trademarks copyrights or other intellectual property rights.

No part of this document may be communicated, distributed, reproduced or transmitted in any form or by any means, electronic or mechanical or otherwise, for any purpose, without the prior written permission of Samsung Electronics Co. Ltd.

The document is subject to revision without further notice.

All brand names and product names mentioned in this document are trademarks or registered trademarks of their respective owners.

For more information, please visit http://developer.samsung.com/