

Look Edge

Programming Guide

Version 1.4.0

Table of Contents






1. OVERVIEW	3
1.1. ARCHITECTURE	5
1.2. CLASS DIAGRAM	6
1.2.1. <i>Slook</i>	6
1.2.2. <i>Edge</i>	7
1.3. SUPPORTED PLATFORMS	8
1.4. SUPPORTED FEATURES	8
1.5. COMPONENTS	8
1.6. INSTALLING THE PACKAGE FOR ANDROIDSTUDIO	8
2. USING LOOK EDGE	10
2.1. USING THE INITIALIZE() METHOD	10
2.2. HANDLING SSDKUNSUPPORTEDEXCEPTION	11
2.3. CHECKING THE AVAILABILITY OF LOOK EDGE PACKAGE FEATURES	11
3. EDGE	12
3.1. CHOOSE EDGE MODE	12
3.1.1. <i>Edge Single Mode</i>	12
3.1.2. <i>Edge Single Plus Mode</i>	12
3.1.3. <i>Edge Feeds Mode</i>	12
3.1.4. <i>Edge Immersive Mode</i>	13
3.2. CHECKING FEATURE	13
3.3. IMPLEMENTING EDGE SINGLE MODE, EDGE SINGLE PLUS MODE OR EDGE FEEDS MODE	14
3.3.1. <i>Declaring a Cocktail Provider in the Manifest</i>	14
3.3.2. <i>Adding CocktailProviderInfo Metadata</i>	15
3.3.3. <i>Implementing a class extending SlookCocktailProvider</i>	16
3.3.4. <i>Updating the RemoteViews with SlookCocktailManager instance</i>	17
3.4. IMPLEMENTING EDGE IMMERSIVE MODE	20
3.4.1. <i>Adding a SubContentView in sub-window</i>	20
3.5. CHECKING THE LANDSCAPE LAYOUT	21
3.6. MINSDK FOR EDGE SINGLE PLUS MODE	22
3.7. CATEGORIES FOR EDGE SPECIALS IN GALAXYAPPS	22
COPYRIGHT	23

1. Overview

Look offers specialized widgets that extend the Android View System to make it more usable, visible, and intuitive.

Look supports the following features:

- **Edge**
 - **Edge Single Mode** is a stand-alone view on the Edge(Curved) screen area [Figure 1].
 - **Edge Single Plus Mode** is a stand-alone view and provides many contents via wide UI. [Figure 1].
 - **Edge Feeds Mode** is similar with Edge Single Mode, but with simpler information [Figure 1].
 - **Edge Immersive Mode** is a mode where the main activity uses Edge(Curved) screen area as sub-window [Figure 1].

Edge Screen Style	Mode	e.g
 <p>Overlay screen style</p>	 <p>Edge Single</p>	 <p>Edge Single</p>
	 <p>Edge Single Plus</p>	 <p>Edge Single Plus</p>

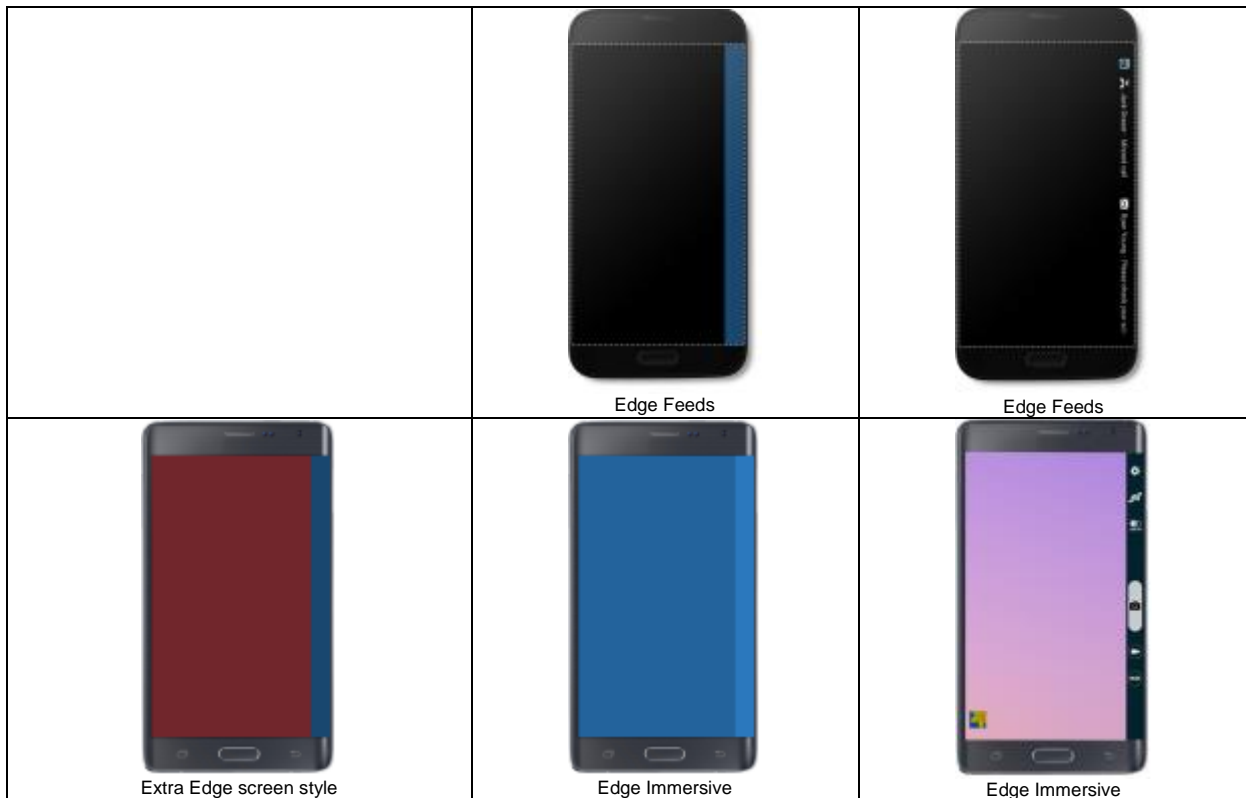


Figure 1: Edge Mode

- **Overlaid Edge Screen Style:** Device with a main screen like a normal device.
- **Extra Edge Screen Style:** Device with a main screen and an extra screen.

1.1. Architecture



Figure 2: Look architecture

The Look architecture consists of:

- **Applications**: One or more applications that use Look.
- **SLookSDK**: Look UI components.
- **View System**: Android Framework View System.

1.2. Class Diagram

1.2.1. Slook

Figure 3 below are the Look classes and interfaces that can be used in your application.

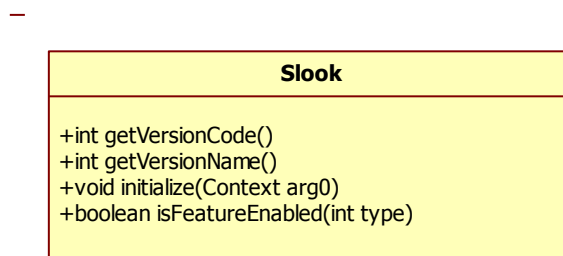


Figure 3: Look classes and interfaces

The Look classes and interfaces include:

- **Slook:** Initializes the Look package.

1.2.2. Edge

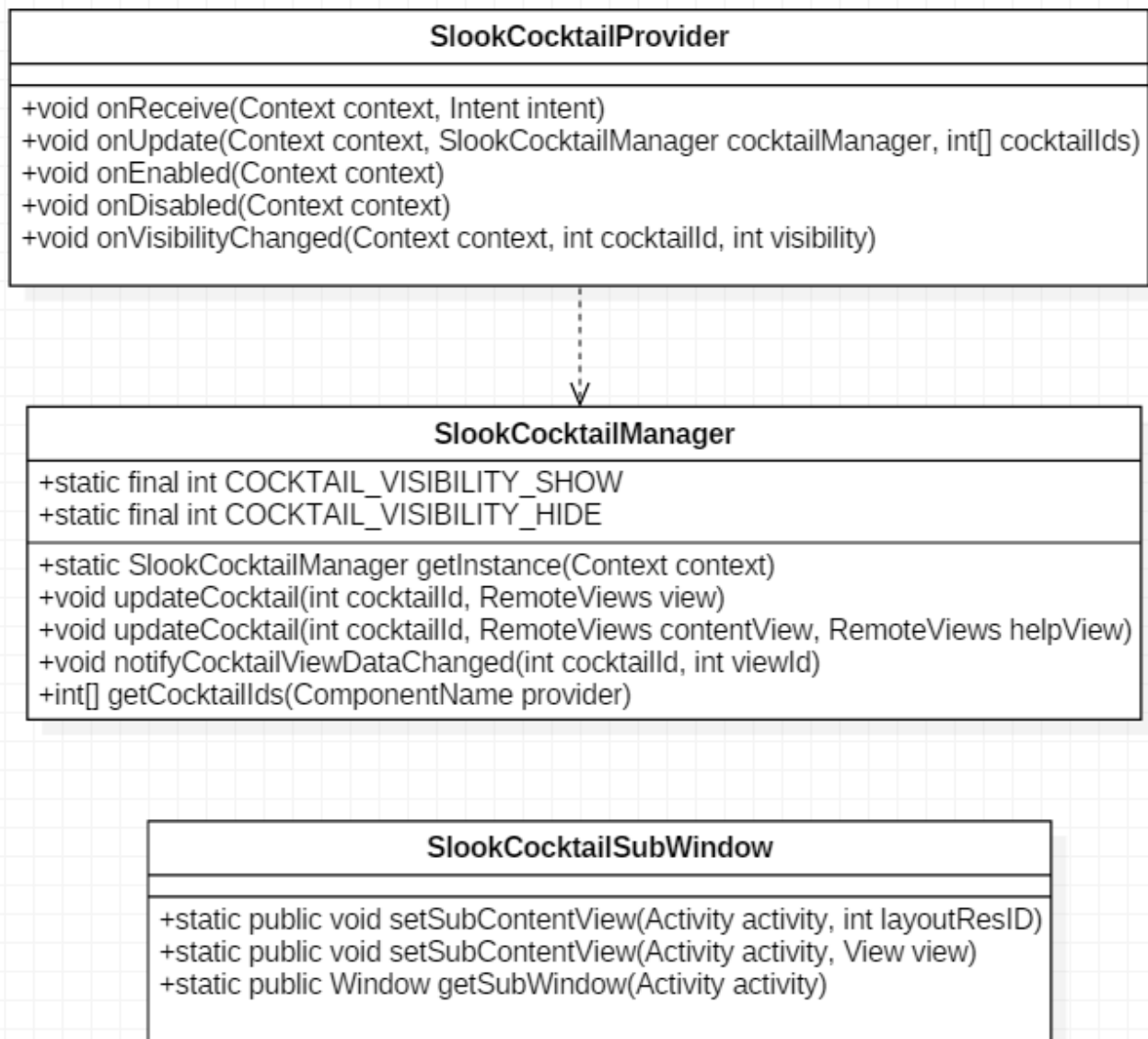


Figure 4: Edge classes

- **SlookCocktailProvider:** A convenience class to aid in implementing a Cocktail provider.
- **SlookCocktailManager:** Updates the cocktail's state and gets information about the installed Cocktail provider.
- **SlookCocktailSubWindow:** Sets a view in the sub-window for Edge immersive mode and gets the sub-window.

1.3. Supported Platforms

The package uses a static Java library that depends on internal Android framework modules. It means that this package only runs on devices that support those modules. Some Samsung Smart Devices do not support Look Edge as it requires devices of the edge series.

1.4. Supported Features

Look Edge supports the following feature:

- **Edge:** Application views on the edge screen as Edge Single mode, Edge Single Plus mode, Edge Feeds mode or Edge Immersive mode.

1.5. Components

- Components
 - look-v1.4.0.jar
 - sdk-v1.0.0.jar
- Imported packages:
 - com.samsung.android.sdk.look

1.6. Installing the Package for AndroidStudio

To install Look for AndroidStudio:

- a.) Add the look-v1.4.0.jar and sdk-v1.0.0.jar files to the libs folder in AndroidStudio.

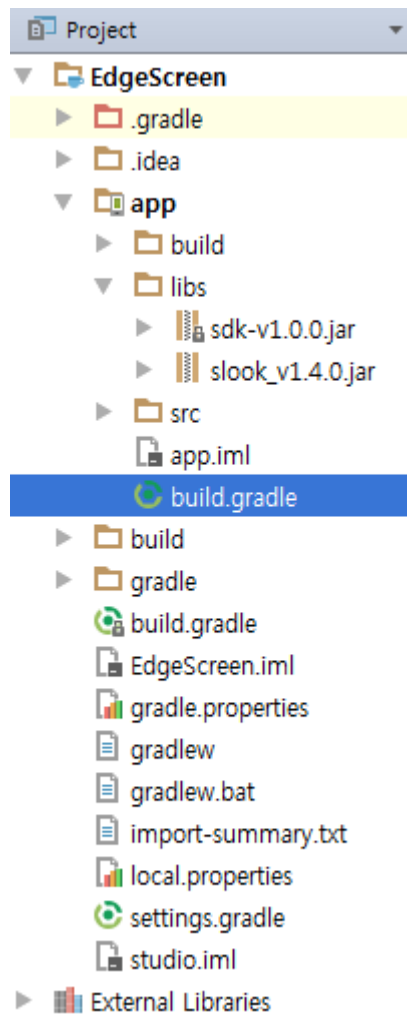


Figure 5: Sample Project in AndroidStudio

b.) Add dependencies for the SDK library folder in build.gradle

```
dependencies{
    compile fileTree(include: '*.jar', dir: 'libs')
    ...
}
```

c.) The following permission has to be specified in the AndroidManifest.xml file to initialize Look.

```
<uses-permission android:name=
"com.samsung.android.providers.context.permission.WRITE_USE_APP_FEATURE_SURVEY"/>
```

If this permission is not added:

- o Android 4.4.2 (KitKat) and above: SecurityException is thrown and your application will not work.
- o Prior to Android 4.4.2 (KitKat): There is no Exception and the application will work properly.

2. Using Look Edge

You need to initialize Slook before it can be used.. Samsung Mobile SDK provides a base class with an `initialize()` method for each package.

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

You can use the `initialize()` method to initialize Slook and also to check if the device supports Slook. If the device does not support Slook or fails to initialize, the method throws an `SsdkUnsupportedException` exception. Handle this exception with the following: If an `SsdkUnsupportedException` exception is thrown, 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 and does not support Slook, the exception type is `SsdkUnsupportedException.DEVICE_NOT_SUPPORTED`.

The Slook class provides the following methods:

- `initialize()` initializes Look. You need to initialize the Look package before you can use it. If the device does not support Look, `SsdkUnsupportedException` exception is thrown.
- `getVersionCode()` gets the Look version number as an integer.
- `getVersionName()` gets the Look version name as a string.
- `isFeatureEnabled(int type)` checks if a Look package feature is available on the device.

```
Slook slook = new Slook();
LinearLayout l = (LinearLayout) findViewById(R.id.information);

try {
    slook.initialize(this);
} catch (SsdkUnsupportedException e) {
    l.addView(createTextView(e.toString()));
    return;
}
```

2.1. Using the initialize() Method

The `Slook.initialize()` method:

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

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

2.2. Handling SsdkUnsupportedException

If an `SsdkUnsupportedException` exception is thrown, check the exception message type using `SsdkUnsupportedException.getType()`.

The following two types of exception messages are defined in the Slook class:

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

2.3. Checking the Availability of Look Edge Package Features

You can check if a Look package feature is supported on the device using the `isFeatureEnabled()` method. The feature types are defined in the Slook class. The feature type is passed as a parameter when calling the `isFeatureEnabled()` method. The method returns a boolean value that indicates the support for the feature on the device.

```
Boolean isFeatureEnabled(int type);
```

The following types are defined as constants in the Slook class:

- Edge Screen
 - COCKTAIL_BAR
 - COCKTAIL_PANEL

3. Edge

3.1. Choose Edge Mode

3.1.1. Edge Single Mode

- Edge Single mode which operates as a stand-alone
- The Edge single mode will be added in the list of the Edge(Curved) Screen setting, and you can enable or disable it.
- If the Edge single mode is enabled, you can use it in the Edge(Curved) screen while using an application in the main screen.
- Please refer to the Edge Single Mode guide for implementing an Edge single mode.
- For this mode, you have to use the COCKTAIL_PANEL feature.

3.1.2. Edge Single Plus Mode

- Edge single plus mode is similar with Edge Single mode, but can provide many contents via the wide UI.
- For the variable width of Edge Single plus mode, Look SDK supports cocktailWidth attribute in cocktail-provider (refer to 3.3.2).
- In order to use Edge Single plus mode, you should define `"com.samsung.android.cocktail.v2.action.COCKTAIL_UPDATE"` as the intent-filter of Provider (refer to 3.3.1).
- For this mode, you have to use the COCKTAIL_PANEL feature.
- For compatibility, be certain to declare "23"(Android 6.0) as API Level in the minSdkVersion attribute (refer to 3.6)

3.1.3. Edge Feeds Mode

- Edge Feeds mode is another kind of Edge Single mode.
- If you want to suggest simple information via Edge Single mode, it is recommended to use the Edge Feeds mode. News, Sports, or Finance are examples.
- If you want to implement an Edge Feeds mode, you should define 'feeds' in the category attribute of the cocktail-provider (refer to 3.3.2).
- Implementation is similar with Edge Single mode.
- Edge feeds mode can be shown both in extra edge screen or overlaid style.
- Edge Feeds mode can be activated during screen off (for Overlaid Edge Device).
- For this mode, you have to use the COCKTAIL_PANEL feature, too.

3.1.4. Edge Immersive Mode

- If a device has an extra edge screen like Galaxy Note Edge, you can add sub-view into sub-window. That means you can use the edge window as sub-window for an activity.
- If you want the implementation that has an extra edge screen showing extra views for the main activity, you should refer the Edge Immersive Mode guide.
- For this mode, you have to use the COCKTAIL_BAR feature.
- It is exclusively supported for devices with extra edge screen like Galaxy Note Edge.

3.2. Checking feature

To check whether a device supports Edge Single Mode, Edge Single Plus Mode or Edge Feeds Mode, you should use the Slook.COCKTAIL_PANEL feature.

If you want to check whether a device has an extra edge screen and supports Edge Immersive Mode, you should use the Slook.COCKTAIL_BAR feature.

The code snippet below checks whether the device supports Edge.

```
Slook slook = new Slook();

try {
    slook.initialize(this);
} catch (SsdkUnsupportedException e) {
    return;
}

if (slook.isFeatureEnabled(Slook.COCKTAIL_PANEL)) {
    // The Device supports Edge Single Mode, Edge Single Plus Mode, and Edge Feeds
    Mode.
}

if (slook.isFeatureEnabled(Slook.COCKTAIL_BAR)) {
    // The Device supports Edge Immersive Mode feature.
}
```

3.3. Implementing Edge Single Mode, Edge Single Plus Mode or Edge Feeds Mode

3.3.1. Declaring a Cocktail Provider in the Manifest

In order to add an Edge Single Mode, Edge Single Plus Mode or Edge Feeds Mode in the Edge(curved) screen area, you need to add the receiver and meta-data for your cocktail provider in the AndroidManifest.xml file.

3.3.1.1 Edge Single Mode or Edge Feeds Mode

You should define `"com.samsung.android.cocktail.action.COCKTAIL_UPDATE"` in the `<action>` element of the `<intent-filter>` element.

```
/AndroidManifest.xml

<receiver android:name=".EdgeSingleProvider" >
    <intent-filter>
        <action android:name=
            "com.samsung.android.cocktail.action.COCKTAIL_UPDATE" />
    </intent-filter>
    <meta-data
        android:name="com.samsung.android.cocktail.provider"
        android:resource="@xml/edg_single">
    </receiver>
```

3.3.1.2 Edge Single Plus Mode

You should define `com.samsung.android.cocktail.v2.action.COCKTAIL_UPDATE` in the `<action>` element of the `<intent-filter>` element.

```
/AndroidManifest.xml

<receiver android:name=".EdgeSinglePlusProvider" >
    <intent-filter>
        <action android:name=
            "com.samsung.android.cocktail.v2.action.COCKTAIL_UPDATE" />
    </intent-filter>
    <meta-data
        android:name="com.samsung.android.cocktail.provider"
        android:resource="@xml/edge_single_plus">
    </receiver>
```

The `<receiver>` element requires the `android:name` attribute which specifies the SlookCocktailProvider used by the Edge Single Mode, Edge Single Plus Mode or the Edge Feeds Mode.

The `<intent-filter>` element must include an `<action>` element with the `android:name` attribute. This attribute specifies that SlookCocktailProvider accepts the ACTION_COCKTAIL_UPDATE broadcast. This is the only broadcast that you must explicitly declare.

The `<meta-data>` element specifies the `CocktailProviderInfo` resource and requires the following attributes:

- ✓ **android:name** – specifies the metadata name. Use [com.samsung.android.cocktail.provider](#) to identify the data as the `CocktailProviderInfo` descriptor.
- ✓ **android:resource** – specifies the `CocktailProviderInfo` resource location.

3.3.2. Adding CocktailProviderInfo Metadata

The `CocktailProviderInfo` defines the essential qualities of Edge Single Mode, Edge Single Plus Mode or Edge Feeds Mode, such as its label, its description, its preview image, category, how often to update it, and a configuration Activity to launch in the Edge(curved) screen setting. Define the `CocktailProviderInfo` object in an XML resource using a single `<cocktail-provider>` element and save it in the project's `res/xml/` folder.

```
<?xml version="1.0" encoding="utf-8"?>
<cocktail-provider xmlns:android=http://schemas.android.com/apk/res/android
    label="@string/edge_name"
    decription="@string/edge_description "
    previewImage="@drawable/edge_preview"
    updatePeriodMills="1800000"
    permitVisibilityChanged="true/false"
    configure="com.example.edge.EdgeConfigure"
    category="feeds"
    cocktailWidth="550"
    launchOnClick="com.example.edge.MainActivity">
</cocktail-provider>
```

Here's a summary of `<cocktail-provider>` attribute:

- ✓ **label** – specifies the name of the Edge Single Mode, Edge Single Plus Mode, or the Edge Feeds Mode.
- ✓ **description** – specifies the description of the the Edge Single Mode, Edge Single Plus Mode, or the Edge Feeds Mode.
- ✓ **previewImage** – shows a preview of what the the Edge Single Mode, Edge Single Plus Mode, or the Edge Feeds Mode will look like after being enabled. It is used to enable or disable Edge Single Mode, Edge Single Plus Mode, or Edge Feeds Mode in the Edge(curved) screen setting.
- ✓ **updatePeriodMillis** – defines how often the Edge Framework should request an update from the `SlookCocktailProvider` by calling the `onUpdate()` callback method. The actual update is not guaranteed to occur exactly on time with the defined value and it is not suggested to update it frequently - not more than once an hour - to conserve the battery. You can also allow the user to adjust the frequency in a configuration.
- ✓ **permitVisibilityChanged** – if it is set to true, the `onVisibilityChanged` of the `SlookCocktailProvider` is called when the Edge Single Mode, Edge Single Plus Mode, or the Edge Feeds Mode is shown.
- ✓ **configure** – defines the Configuration Activity for the Edge Single Mode, Edge Single Plus Mode or the Edge Feeds Mode, which can be launched in the Edge(curved) screen setting.
- ✓ **category** – defines the category of the Edge Feeds Mode. There are 'feeds'. The specific devices will support only the Edge Feeds Mode defined in the specified category.
- ✓ **cocktailWidth** – defines the width of Edge Single Plus Mode. If you use the attribute, your provider should use [com.samsung.android.cocktail.v2.action.COCKTAIL_UPDATE](#) as the intent filter.

- ✓ **launchOnClick** – defines the button of Edge Single Plus Mode. You can set a button on Edge Single Plus Mode, launching the main activity you want. If you use the attribute, your provider should use `com.samsung.android.cocktail.v2.action.COCKTAIL_UPDATE` as the intent filter.

3.3.3. Implementing a class extending SlookCocktailProvider

The SlookCocktailProvider class extends BroadcastReceiver as a convenience class to handle the Edge Broadcasts.

```
package com.example.edgescreen.provider;

import com.samsung.android.sdk.look.cocktailbar.SlookCocktailManager;
import com.samsung.android.sdk.look.cocktailbar.SlookCocktailProvider;

.....

public class CocktailSampleProvider extends SlookCocktailProvider {
    @Override
    public void onUpdate(Context context, SlookCocktailManager
        cocktailBarManager, int[] cocktailIds) {
        .....
    }
}
```

onUpdate(Context, SlookCocktailManager, in[])

This method is called when the user adds the Edge Single Mode, Edge Single Plus Mode or the Edge Feeds Mode. If necessary, it should perform the essential setup, such as defining event handlers for views and starting a temporary Service.

onEnabled(Context)

This method is called when the Edge Single Mode, Edge Single Plus Mode or the Edge Feeds Mode is created for the first time. If you need to open a new database or perform setup, then this is a good place to do it.

onDisabled(Context)

This method is called when the instance of your Edge Single Mode, Edge Single Plus Mode or Edge Feeds Mode is deleted from the enabled list. This is where you should clean up any work done in onEnabled(Context), such as delete a temporary database.

onReceive(Context, Intent)

This method is called for every broadcast and before each of the above callback methods.

onVisibilityChanged(Context, int, int)

This method is called when the visibility of the Edge Single Mode, Edge Single Plus Mode or the Edge Feeds Mode is changed.

If the visibility is COCKTAIL_VISIBILITY_SHOW, the state of the Edge Single Mode, Edge Single Plus Mode or the Edge Feeds Mode is shown. At that time, if you want to refresh your Edge Single Mode, Edge Single Plus Mode or Edge Feeds Mode, you should call updateCocktail with a new RemoteViews. If you want the callback method called, you have to define `permitVisibilityChanged` to `true` in the CocktailProviderInfo (refer to section 3.3.2)

3.3.4. Updating the RemoteViews with SlookCocktailManager instance

```
RemoteViews rv = new RemoteViews(context.getPackageName(), R.layout.edge_panel);
setPendingIntent(context, rv);
for (int i = 0; i < cocktailIds.length; i++) {
    cocktailBarManager.updateCocktail(cocktailIds[i], rv);
}
```

You must define an initial layout for your Edge Single Mode, Edge Single Plus Mode or Edge Feeds Mode in an XML and save it in the project's res/layout/ directory.

Creating the Edge Single Mode, Edge Single Plus Mode, or the Edge Feeds Mode layout is simple if you're familiar with Layouts. However, you must be aware that the Edge Single Mode, Edge Single Plus Mode, and the Edge Feeds Mode layouts are based on RemoteViews, which do not support all kinds of layout or view widgets.

The RemoteViews object can support the following layout classes:

- FrameLayout
- LinearLayout
- RelativeLayout,
- GridLayout
- Button
- ImageButton
- ImageView
- ProgressBar
- TextView
- ViewFlipper
- ListView.

Please refer to this URL for more information on [RemoteViews](#).

For GUI performance [RemoteViews.reapply](#) API is used when update RemoteViews was already inflated.

3.3.4.1 SlookCocktailManager APIs

static SlookCocktailManager getInstance(Context)

Get the SlookCocktailManager instance to use for the supplied context object.

void updateCocktail(int id, RemoteViews view)

Set the Remoteviews to use for the specified cocktail ID.

void updateCocktail(int id, RemoteViews contentView, RemoteViews helpView)

Set the content view and help view to use for the specified cocktail ID.

The Help view object can support basic widget classes, which does not support RemoteService:

- FrameLayout
- LinearLayout
- RelativeLayout
- GridLayout
- Button
- ImageButto
- ImageView
- ProgressBar
- TextView
- ViewFlipper.

void notifyCocktailViewDataChanged(int id, int viewId)

Notifies the specified collection view in the specified Cocktail instance.

int[] getCocktailIds(ComponentName provider)

Get the cocktailIds that have been bound to the given Cocktail provider.

void setOnPullPendingIntent(int id, int viewId, PendingIntent pendingIntent)

Set pull to refresh interaction for specified view to notify invalidate their data.

void setOnLongClickPendingIntent(RemoteViews rv, int viewId, PendingIntent longClickPendingIntent)

Set long click pending intent to specified view in RemoteViews.

void setOnLongClickPendingIntentTemplate(RemoteViews rv, int viewId, PendingIntent pendingIntentTemplate)

Set long click pending intent template to specified collections in RemoteViews.

3.3.4.2 Use pull to refresh interaction

From Slook SDK v1.4.0, you can easily add pull to refresh interaction on your edge panel by setOnPullPendingIntent API. After set pull to refreshing target view, it will be added on SwipeRefreshLayout and it send registered pendingIntent with pulling gesture on target view. The target view of SetOnPullPendingIntent API is available for panel contentView not for panel helpView.

```
Intent refreshintent = new Intent(ACTION_PULL_TO_REFRESH);
PendingIntent pendingIntent = PendingIntent.getBroadcast(context, 0xff,
refreshintent, PendingIntent.FLAG_UPDATE_CURRENT);
SlookCocktailManager.getInstance(context).setOnPullPendingIntent(cocktailIds[0],
R.id.remote_list, pendingIntent);
```

3.3.4.3 Set long click pending intent

Similar with RemoteViews.SetOnClickPendingIntent API you can set a pending intent to launch with long clicking on your edge panel. This is supported from SDK v1.4.0.

```
RemoteViews stateView = new RemoteViews(context.getPackageName(),
R.layout.long_click_state_layout);
SlookCocktailManager.getInstance(context).setOnLongClickPendingIntent(stateView,
R.id.state_btn1, pendingIntent);
```

If long click target view using collections (eg. ListView, GridView etc.) you can set a pendingIntent template instead set them on each items of collections individually.

```
RemoteViews remoteListView = new RemoteViews(context.getPackageName(),
R.layout.remote_list_view);
SlookCocktailManager.getInstance(context).setOnLongClickPendingIntentTemplate(remoteLi
stView, R.id.remote_list, templatePendingIntent);
```

To work pendingIntent template you need to set fillnIntent on its item views, and it should set on root view of item layout.

```
private class SampleRemoveViewFactory implements
```

```
RemoteViewsService.RemoteViewsFactory {
    @Override
    public RemoteViews getViewAt(int id) {
        RemoteViews itemView = new RemoteViews(getPackageName(),
            R.layout.remote_list_item);
        ...
        itemView.setOnClickFillInIntent(R.id.item_root, intent);
        return itemView;
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/item_root"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    ...
</FrameLayout>
```

setOnLongClickPendingIntentTemplate works base on AdapterView.setOnItemLongClickListener. It could be restricted on some collections widgets such as StackView and AdapterViewFlipper.

3.4. Implementing Edge Immersive Mode

3.4.1. Adding a SubContentView in sub-window

If you want to use the Edge(curved) screen area as a sub-window for an Activity, you should set a layout resource or a view into the sub-window with SlookCocktailSubWindow class and add a meta-data for the sub-window in the AndroidManifest.xml file.

```
public class ImmersiveModeActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Slook slook = new Slook();
        try {
            slook.initialize(this);
        } catch (SdkUnsupportedException e) {
            return;
        }
        if (slook.isFeatureEnabled(Slook.COCKTAIL_BAR)) {
            /* If device supports Edg Immersive Mode, you can set up
               the sub-window. */
            setContentView(R.layout.expaneded_activity_main);
            SlookCocktailSubWindow.setSubContentView(this, R.layout.sub_view);
        } else {
            // Normal device.
            setContentView(R.layout.activity_main);
        }
    }
}
```

```

/AndroidManifest.xml
<activity
.....
<meta-data
    android:name="com.samsung.android.cocktail.subwindow.enable"
    android:value="true" />

</activity>

```

3.5. Checking the landscape layout for feeds mode

Feeds mode works not only vertical mode but also horizontal mode. Due to orientation changes, you must add the layout resources for horizontal mode. If you don't make layout resources for horizontal mode, the vertical layout will be shown in landscape mode that might create critical defects.

```

/res/layout/sample_panel.xml

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="120dp"
        android:background="@android:color/hoLo_orange_light"
        android:orientation="vertical" />

```

```

/res/layout-land/sample_panel.xml

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="horizontal" >

    <LinearLayout
        android:layout_width="120dp"
        android:layout_height="match_parent"
        android:background="@android:color/hoLo_orange_light"
        android:orientation="horizontal" />

```

3.6. MinSdk for Edge Single Plus Mode

Edge Single Plus Mode is supported on Android 6.0 and above. Therefore, set your minSdkVersion to “23” to identify the lowest API level with which your app is compatible.

```
/AndroidManifest.xml  
  
<uses-sdk  
    android:minSdkVersion="23"  
    android:targetSdkVersion="23" />
```

3.7. Categories for Edge specials in GalaxyApps

Metadata should be defined in AndroidManifest.xml to register Edge specials in GalaxyApps.

The `<meta-data>` element specifies the search filter which requires the following attributes:

- ✓ **android:name** – specifies the metadata name. Use `com.samsung.android.cocktail.mode` to identify the categories on the GalaxyApps.
- ✓ **android:resource** – specifies the Edge Mode for GalaxyApps

Metadata Value	Description
<code>edge_single</code>	Supports Edge Single Mode
<code>edge_immersive</code>	Supports Edge Immersive Mode
<code>edge_feeds</code>	Supports Edge Feeds Mode
<code>Edge_single_plus</code>	Supports Edge Single Plus Mode

```
/AndroidManifest.xml  
  
<meta-data  
    android:name="com.samsung.android.cocktail.mode"  
    android:value="edge_single_plus" />
```

Copyright

Copyright © 2015 Samsung Electronics Co. Ltd. All Rights Reserved.

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/>