MultiWindow

Programming Guide

Version 1.0

Table of Contents

1. Overview

MultiWindow allows you to run multiple resizable applications simultaneously.

MultiWindow consists of the MultiWindow UI and MultiWindow Framework. The MultiWindow UI has a MultiWindow launcher called Traybar. You can use it to manage MultiWindow applications. You can launch the Traybar by long-pressing the **Back** key and use it to run multiple applications. MultiWindow Framework is a system service connected to Activity Manager, Window Manager, and View System to configure applications in MultiWindow.

You can register your application as a MultiWindow application and add it to the Traybar by declaring it in your Android manifest file.

You can use MultiWindow to:

- check the status of your MultiWindow application
- change the status
- check the status change

1.1. Architecture

The following figure shows the MultiWindow architecture.

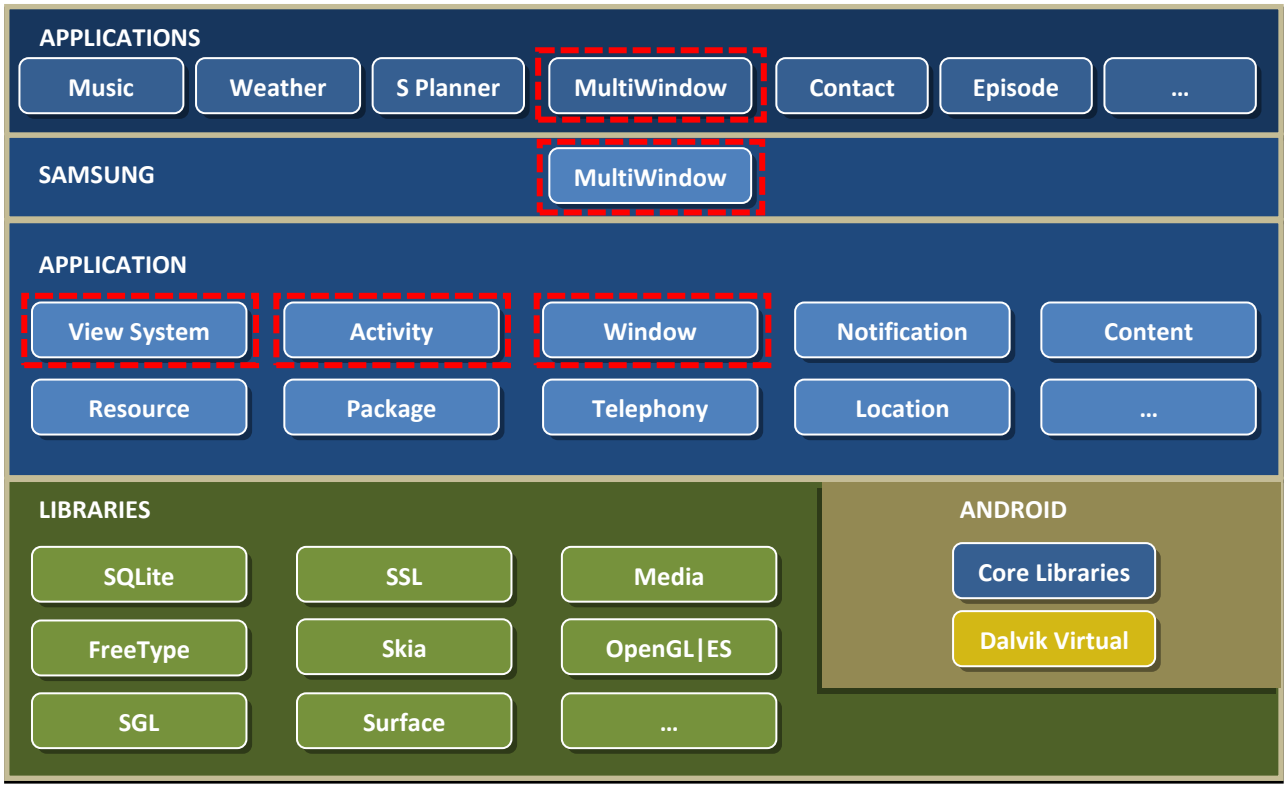

Figure 1: MultiWindow architecture

The architecture consists of:

- **MultiWindow UI:** UI to run MultiWindow applications. It consists of:
    - Traybar: UI component for managing and launching applications.
    - Centerbar window: UI component for separating and resizing applications when two MultiWindow applications are executed simultaneously.
    - Centerbar button: UI component for switching applications, changing the application position and size, or terminating applications.
- **MultiWindow Framework:** Activity Manager, Window Manager, and View System are interlinked with the MultiWindow Framework. The MultiWindow data structures and the system service for algorithm management are in the application framework layer.
- **MultiWindow package:** The system service for MultiWindowManagerService, which controls the MultiWindow data.

1.2. Class Diagram

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

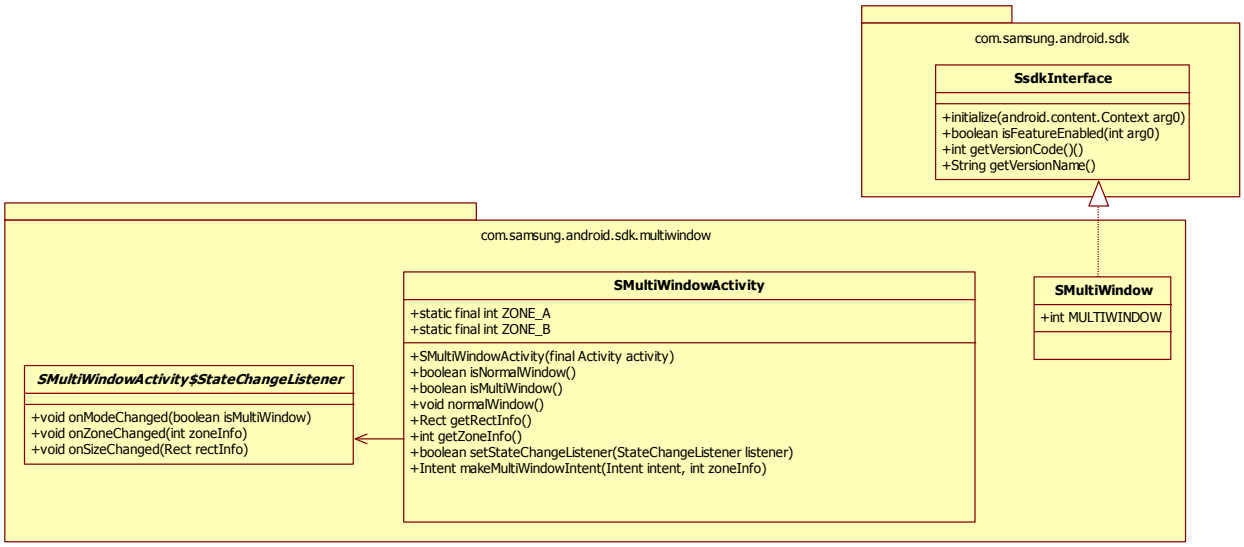

Figure 2: MultiWindow classes and interfaces

The MultiWindow classes and interfaces include:

- **SMultiWindowActivity:** Checks or changes the activity status of MultiWindow.
- **SMultiWindowActivity$StateChangeListener:** Listens for MultiWindow status change events.

1.3. Supported Platforms

Android 4.1(Android API level 16) or above support MultiWindow.

1.4. Supported Features

MultiWindow supports the following features:

- Checking the support of MultiWindow
- Checking the status of MultiWindow
- Listener for MultiWindow status change events

1.5. Components

- Components
  - multiwindow-v1.0.0.jar
- Imported packages:
  - com.samsung.android.sdk.multiwindow.SMultiWindow
  - com.samsung.android.sdk.multiwindow.SMultiWindowActivity

1.6. Installing the Package for Eclipse

To install MultiWindow for Eclipse:

1. Add the multiwindow-v1.0.0.jar and sdk-v1.0.0.jar files to the libs folder in Eclipse.

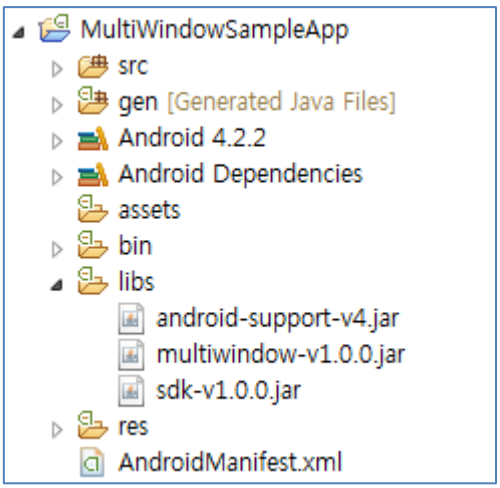

Figure 3: libs folder in Eclipse

2. Hello MultiWindow

The following program changes the size of the screen using MultiWindow.

```
public class HelloMultiWindow extends ListActivity {
	private static final int MENU_MULTI_WINDOW = 0;
	private static final int MENU_NORMAL_WINDOW = 1;

	private SMultiWindow mMultiWindow = null;
	private SMultiWindowActivity mMultiWindowActivity = null;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		// create Creates list item's string. Each item is used for MultiWindow
		// test.
		String[] mStrings = new String[] { "1. Multi Window",
				"2. Normal Window" };

		setListAdapter(new
ArrayAdapter(this,android.R.layout.simple_list_item_1, mStrings));
		getListView().setTextFilterEnabled(true);

		// Creates the MultiWindow instance
		mMultiWindow = new SMultiWindow();

		try {
			mMultiWindow.initialize(this);
		} catch (SsdkUnsupportedException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
			// In this case, application should be stopped, because this
device can't support this feature.
		}

		mMultiWindowActivity = new SMultiWindowActivity(this);
	}

	@Override
	protected void onListItemClick(ListView l, View v, int position, long id) {
		switch (position) {
		case MENU_NORMAL_WINDOW:
			// Change: Multi -> Normal Window
			mMultiWindowActivity.normalWindow();
			break;
		case MENU_MULTI_WINDOW:
			{
				// Change: Normal -> Multi Window
				Intent intent = new Intent(Intent.ACTION_MAIN);
				intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
				intent.setComponent(new
ComponentName("com.sec.android.app.sbrowser",

	"com.sec.android.app.sbrowser.SBrowserMainActivity"));
				SMultiWindowActivity.makeMultiWindowIntent(intent,
SMultiWindowActivity.ZONE_A);
```

```
                        startActivity(intent);
                    }
                    break;
                }
            }
}
```


3. Using the SMultiWindow Class

The SMultiWindow class provides the following methods:

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

```
SMultiWindow mMultiWindow = new SMultiWindow();
try {
      mMultiWindow.initialize(this);
} catch (SsdkUnsupportedException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
}
```

3.1. Using the initialize() Method

The `SMultiWindow.initialize()` method:

- initializes the MultiWindow package
- checks if the device is a Samsung device
- checks if the device supports the MultiWindow package
- checks if the MultiWindow libraries are installed on the device

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

If the MultiWindow package fails to initialize, the `initialize()` method throws an SsdkUnsupportedException exception. To find out the reason for the exception, check the exception message.

3.2. 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 SMultiWindow class:

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

3.3. Checking the Availability of MultiWindow Package Features

You can check if a MultiWindow package feature is supported on the device with the `isFeatureEnabled()` method. The feature types are defined in the SMultiWindow class. Pass the feature type 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);
```

4. Using the MultiWindow Package

4.1. Configuring Your Application for MultiWindow

4.1.1. Making Your Application a MultiWindow Application

To include your application in the MultiWindow UI Traybar, add the following meta-data to your Android manifest file below the application tag.

```
<meta-data android:name="com.samsung.android.sdk.multiwindow.enable"
           android:value="true" />
```

4.1.2. Making Your Application a Multi-Instance Application

To run multiple instances of your application:

- do not declare `launchMode` for an activity component in the launcher category to be single task and single instance
- do not use static member variables in your code except for member variables that share data structures between instances
- add the following meta-data to your Android manifest file below the application tag
- meta-data to declare just will not work, it only works over the tray bar to launch(since note3)
- maximum instances are 2

```
<meta-data android:name="com.samsung.android.sdk.multiwindow.multiinstance.enable"
           android:value="true" />
```

The maximum number of instances allowed for an application is two (2).

4.1.3. Configuring the Pen Window

To enable the Pen Window UI for your application, add the following meta-data to your Android manifest file.

```
<meta-data android:name="com.samsung.android.sdk.multiwindow.penwindow.enable"
           android:value="true" />
```

4.1.4. Configuring your MultiWindow Style

To configure a MultiWindow style for your application, add the following meta-data to your Android manifest file.

```
<meta-data android:name="com.samsung.android.sdk.multiwindow.style"
           android:value="forceTitleBar|fullscreenOnly" />
```

The table below shows the styles that MultiWindow supports.

Table 1: MultiWindow styles

MultiWindow Style	Description
"forceTitleBar"	Make multiwindow titlebar force if the window cannot make titlebar. (ex. Translucent window type)
"fullscreenOnly"	Blocks support for MultiWindow for some activities in applications where MultiWindow is supported

Use the divider '|' to assign two or more styles.

4.1.5. Configuring your Traybar

To configure a Traybar for your application, add the following code. By default, the Traybar is enabled.

```
import android.view.WindowManager;

WindowManager.LayoutParams lp = getWindow().getAttributes();
lp.privateFlags |= WindowManager.LayoutParams.
            PRIVATE_FLAG_DISABLE_MULTI_WINDOW_TRAY_BAR;getWindow().setAttributes(lp);
```

The Traybar is show in the figure below.

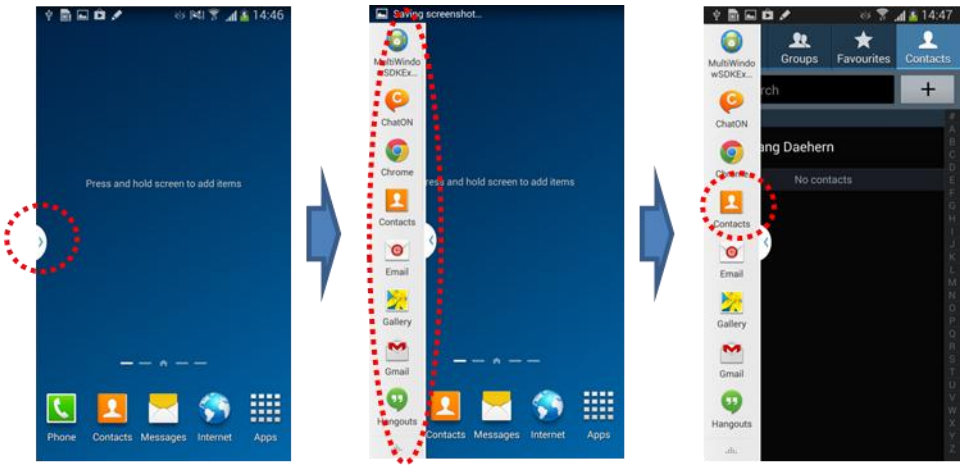

Figure 4: Traybar

4.2. Checking for MultiWindow Support on a Device

To check for MultiWindow support on a device, use the `SMultiWindow.getVersionCode()` method. MultiWindow is supported if the value returned is greater than 0.

```
mMultiWindow = new SMultiWindow();

mMultiWindow = new SMultiWindow();

SMultiWindow mMultiWindow = new SMultiWindow();
if (mMultiWindow.getVersionCode() > 0) {
       // This device support MultiWindow Feature.
}

if (mMultiWindow.isFeatureEnabled(SMultiWindow.MULTIWINDOW)) {
              // This device support MultiWindow.}
}
```

4.3. Running MultiWindow

You can use the `SMultiWindowActivity.makeMultiWindowIntent()` method to call other applications in MultiWindow through startActivity.

The MultiWindow zone info is based on the task record. You have to specify the zone information to execute MultiWindow with Intent.

```
// Executes MultiWindow
String launchType = launchTypes.get(which);
ResolveInfo selectApp = mMultiWindowAppList.get(selectedApp);
ComponentInfo selectAppInfo = selectApp.activityInfo != null ?
      selectApp.activityInfo : selectApp.serviceInfo;

Intent intent = new Intent(Intent.ACTION_MAIN);
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
intent.setComponent(new ComponentName(selectAppInfo.packageName,
                                      selectAppInfo.name));
SMultiWindowActivity.makeMultiWindowIntent(intent, SMultiWindowActivity.ZONE_A);
startActivity(intent);
```

For more information, see `makeMultiWindowIntent()` in MultiWindowDemoFunction.java in MultiWindowSampleApp.

4.3.1. Checking MultiWindow Status Information

You can use the following methods to check the MultiWindow status information:

- `boolean isNormalWindow()`
- `boolean isMultiWindow()`
- `Rect getRectInfo()`
- `int getZoneInfo()`

For more information, see `getDemoResult()` in MultiWindowDemoFunction.java in MultiWindowSampleApp.

4.3.2. Listening to MultiWindow Status Change Events

You can listen to MultiWindow status change events by registering a MultiWindow StateChangeListener instance. Only this API is operated from Android API level 18(Android 4.3) or above version.

- **StateChangeListener:** Called when there are changes in the Mode, Zone, or Size status in MultiWindow.

Listener Method	Description
onModeChanged	Called when the status changes between Normal Window and MultiWindow. The Boolean parameter value indicates the status change.
onZoneChanged	Called when the information on the top/bottom or the left/right changes. The changed zone information is checked with the integer parameter information.
onSizeChanged	Called when the size of the application changes through the Centerbar UI. The changed Rect information is checked with the Rect parameter information.

```
// Registers window state listener

mMultiWindowActivity
              .setStateChangeListener(new SMultiWindowActivity.StateChangeListener() {
              public void onModeChanged(boolean isMultiWindow) {
                     if (isMultiWindow)
                           ;// TO BE IMPLENTED
                           // Normal true --> called when changing to
                           // Multiple Window
                     else
                           ;// TO BE IMPLEMENTED
                           // Multiple false -> called when changing to
                           // Normal Window
              }

              public void onZoneChanged(int arg0) {
                     if (arg0 == SMultiWindowActivity.ZONE_A)
                           ;// TO BE IMPLEMENTED
                     else if (arg0 == SMultiWindowActivity.ZONE_B)
                           ;// TO BE IMPLEMENTED
              }

              public void onSizeChanged(Rect arg0) {

              }
});
```

For more information, see onModeChanged() in MultiWindowDemoFunction.java in MultiWindowSampleApp.

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/