Look

Programming Guide

Version 1.0

Table of Contents

1. Overview

Look offers you specialized widgets to extend the Android View System to make itmore usable, visible, and intuitive.

You can use the following Look features in your application:

- AirButton to add a popup window with menus and buttons.
- SmartClip allows users to delineate a region to capture a screen shot with S-Pen. As a developer, you can provide metadata from your application to SmartClip.
- WritingBuddy(DirectInput) to providehand writing panel pop ups on user input areas for users to write with S-Pen.
- PointerIconto provide more image options for the hovering pointer.

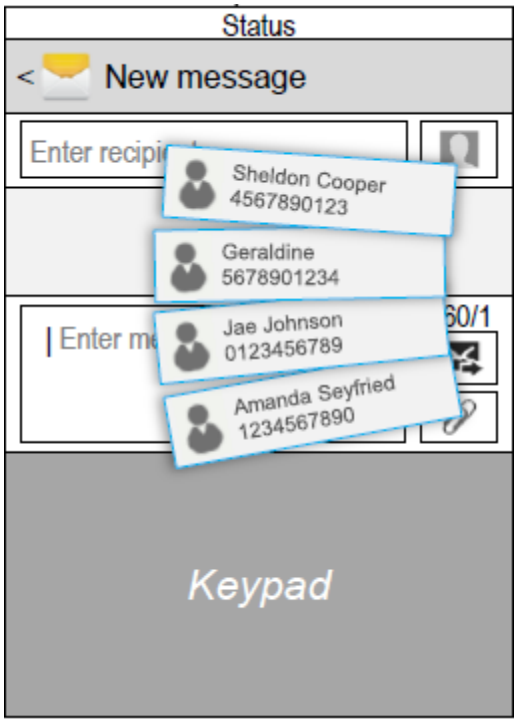

Figure 1: AirButton

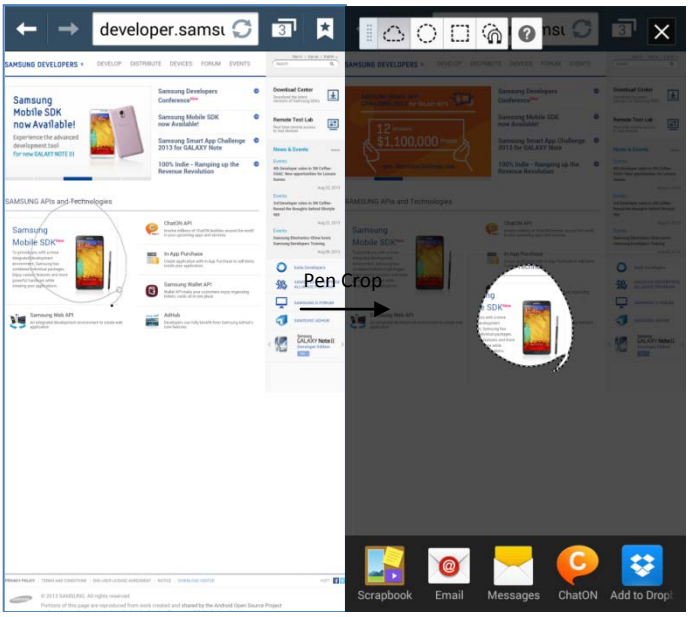

Figure 2: SmartClip

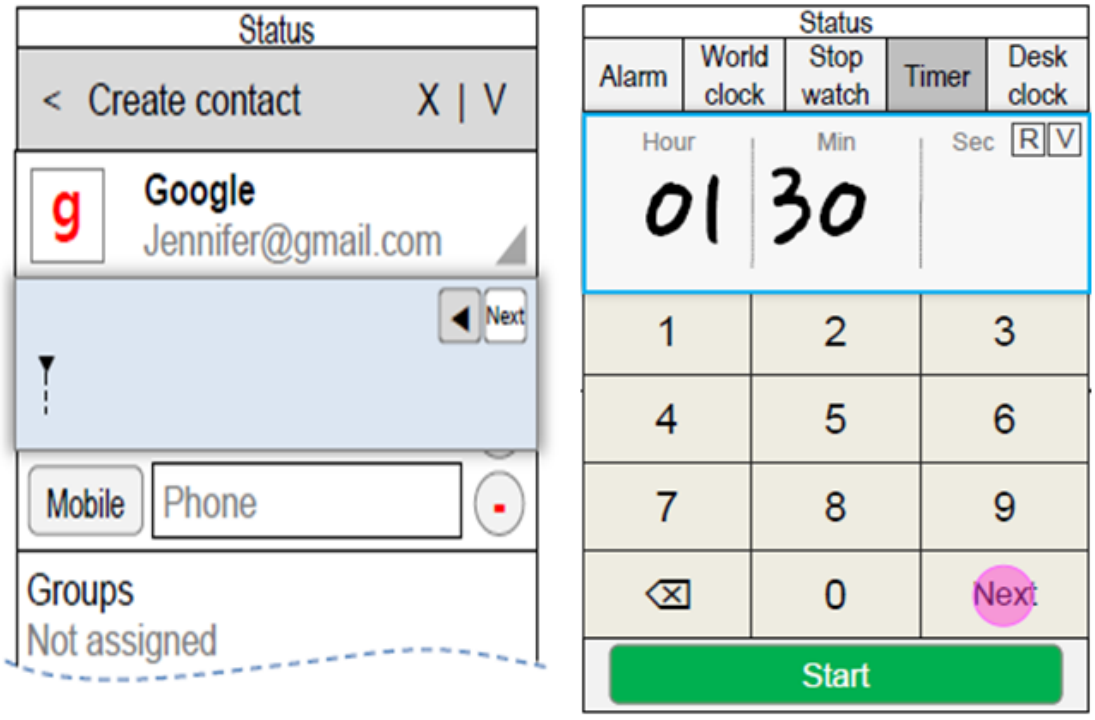

Figure 3: WritingBuddy

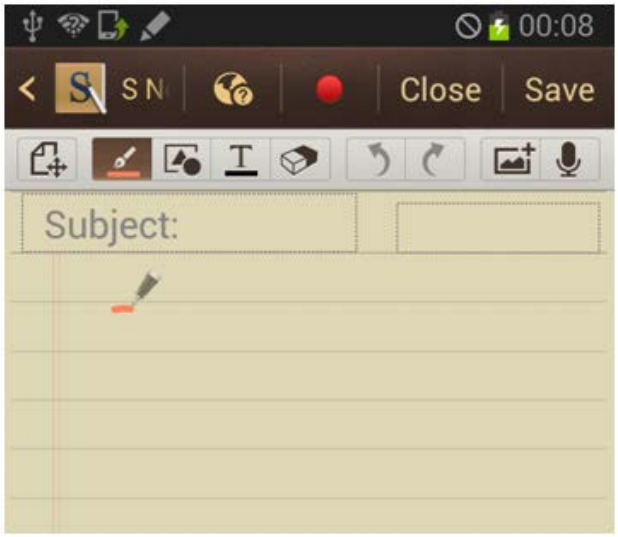

Figure 4: PointerIcon

1.1. Architecture

The following figure shows the Look architecture.

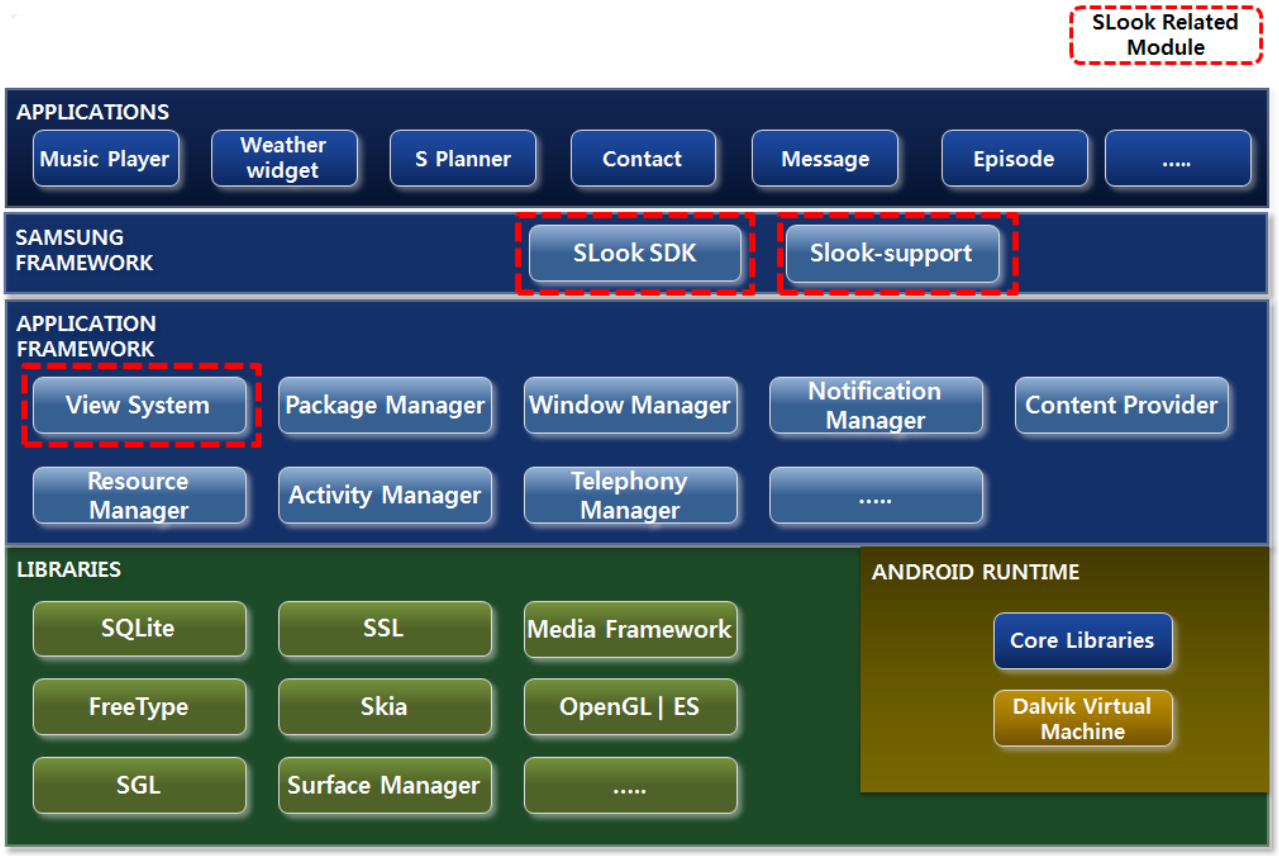

Figure 5: Look architecture

The architecture consists of:

- **Applications:**One or more applications that use Look.
- **SLookSDK:**LookUI components.
- **View System:**AndroidFrameworkView System.

1.2. Class Diagram

The following figures show the Look classes and interfaces that you can use in your application.

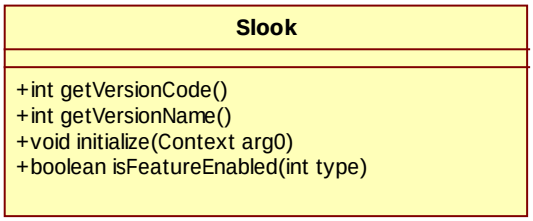

Figure 6: Look classes and interfaces

The Look classes and interfaces include:

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

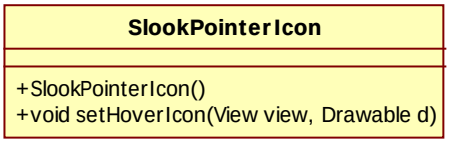

Figure 7: PointerIcon classes

- **SLookPointerIcon:**Defines the image for the hover pointer.

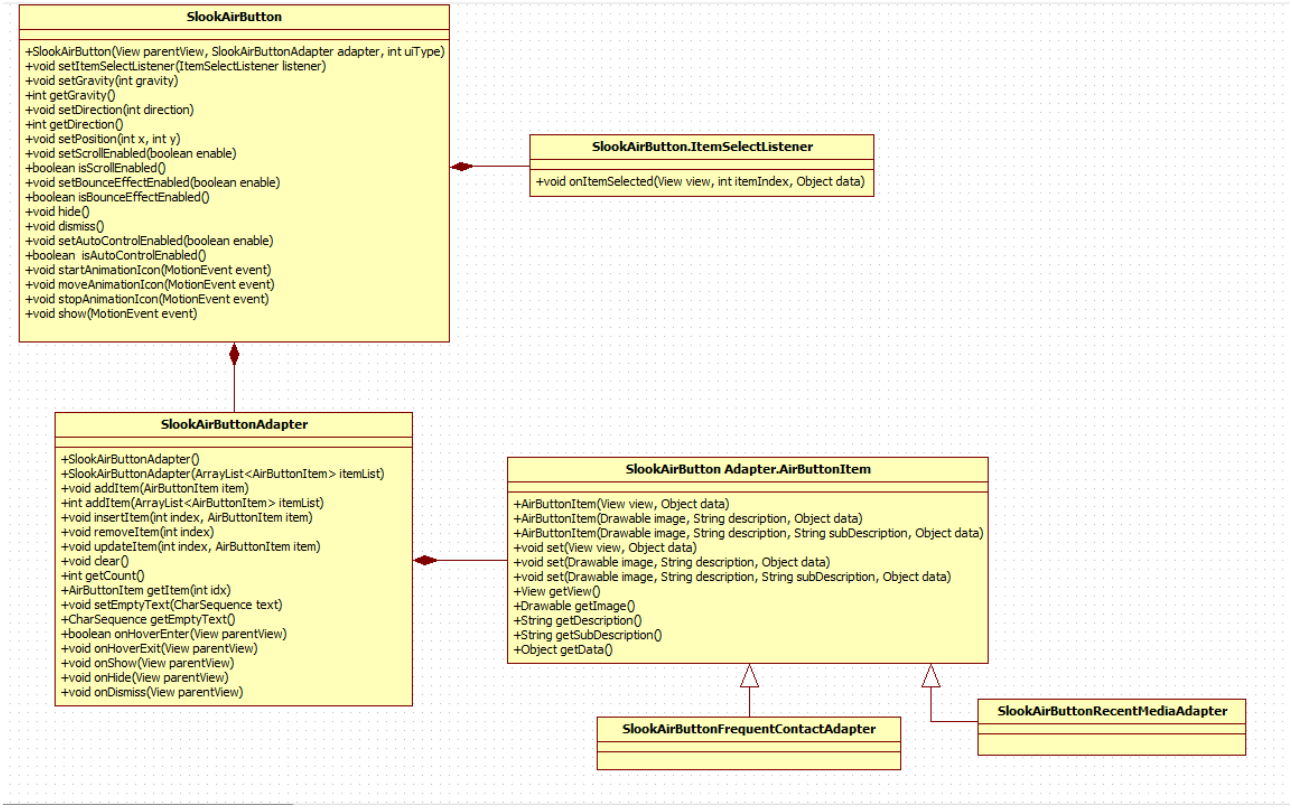

Figure 8: AirButton classes

- **SlookAirButton:** Draws AirButton on View.
- **SlookAirButton.ItemSelectListener:**Listener that is notified when an item on AirButton is selected.
- **SlookAirButtonAdapter:** Manages the content ofAirButton menus.
- **SlookAirButtonAdapter.AirButtonItem:** Contains AirButton item information.
- **SlookAirButtonFrequentContactAdapter:** Displays frequently accessed contacts.
- **SlookAirButtonRecentMediaAdapter:** Displays recently played media.

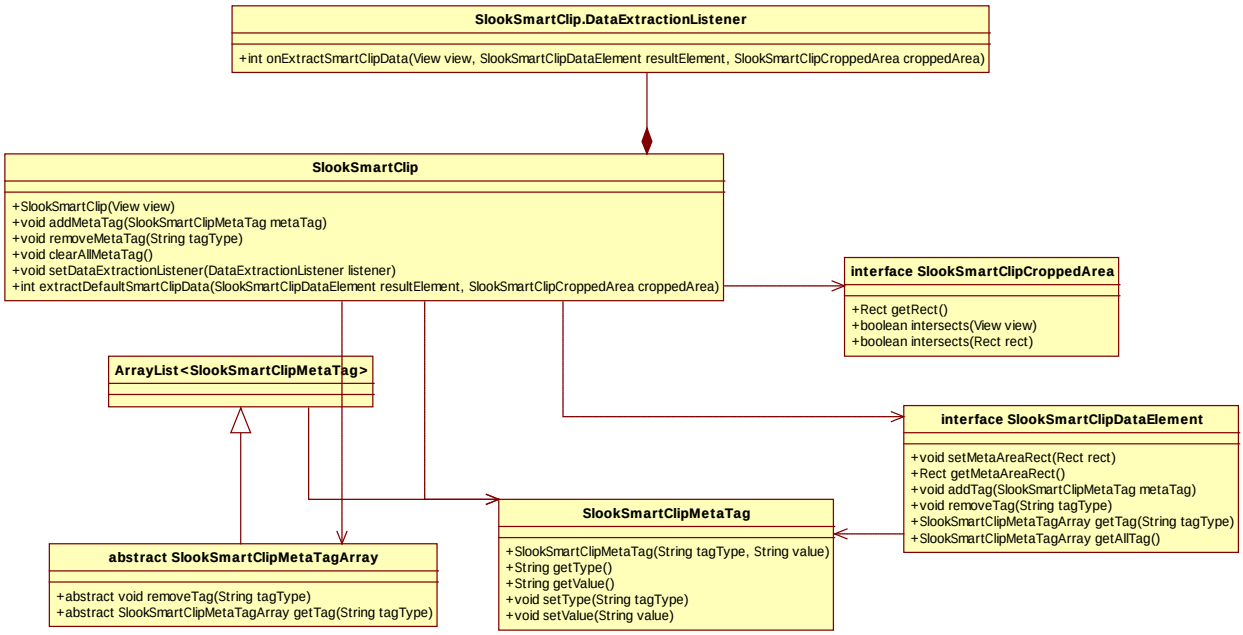

Figure 9: SmartClip classes

- **SlookSmartClip:** Processes the metadata and images from the captured area.
- **SlookSmartClip.DataExtractionListener:**Listener that is notified when users delineate an area for capture.
- **SlookSmartClipCroppedArea:** Manages the captured area.
- **SlookSmartClipDataElement:** Processes the metadata.
- **SlookSmartClipMetaTag:** Defines and manages the meta tags.
- **SlookSmartClipMetaTagArray:** Manages the array of meta tags.

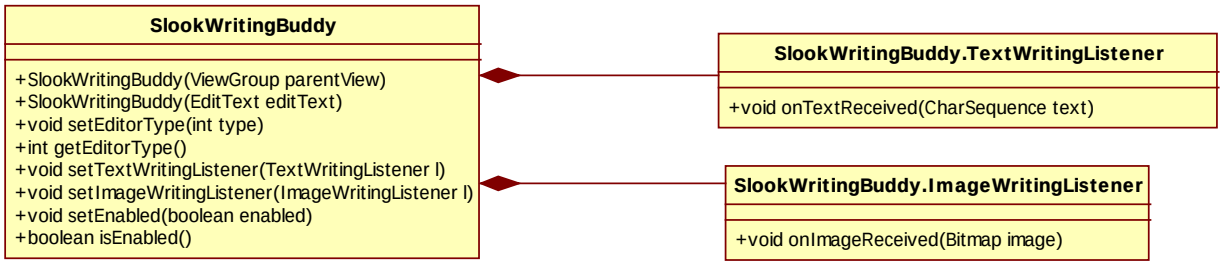

Figure 10: WritingBuddy classes

- **SlookWritingBuddy:**Provides methods for accessing an editor for editing text or drawing images.
- **SlookWritingBuddy.TextWritingListener:**Listens for text editing completion events.
- **SlookWritingBuddy.ImageWritingListener:** Listens for image drawing completion events.

1.3. Supported Platforms

The package uses a static Java library that depends on internal Android framework modules. This means this package only runs on devices that support those modules.

Look requires thatS-Pen be used to access its features.

Some Samsung Smart Devices do not support Look.

1.4. Supported Features

Look supports the following features:

- **AirButton:** Quick access menu for various contexts.
- **SmartClip:** Powerful screenshot capture with metadata recognition.
- **WritingBuddy:** Simple hand writing application.
- **PointerIcon:** More options for hovering pointers.

1.5. Components

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

1.6. Installing the Package for Eclipse

To install Look for Eclipse:

Add theslook-v1.0.0.jar and sdk-v1.0.0.jarfile to the libs folder in Eclipse.

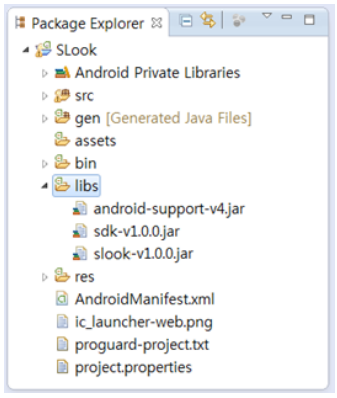

Figure 11: libs folder in Eclipse

2. Using the Slook Class

You need to initialize a Slook before you can use it. 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 a `initialize()` method to initialize it and also to check if the device supports the Slook. If the device does not support the Slook, 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 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 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.

```
Slookslook = newSlook();
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) throwsSsdkUnsupportedException
```

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

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 Package Features

You can check if a Look package feature is supported on the device with the isFeatureEnabled() method. The feature types are defined in the Slook 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.

```
booleanisFeatureEnabled(int type);
```

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

- AIRBUTTON
- SMARTCLIP
- WRITINGBUDDY
- SPEN_HOVER_ICON

3. PointerIcon

PointerIcon provides you more options for the hovering pointer, which appears when S-Pen is close to the viewport of the device. You can use a more intuitive image for an activity or the Pen status. For example, you can use a color palette icon for the hover pointer in your application.

3.1. Hello PointerIcon

Hello PointerIcon is a simple application that:

- creates a PointerIcon instance
- sets an icon as the pointer during hover

Add the following layout in your activity_main.xml file for the sample application.

```
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
>
<Button
android:id="@+id/btn_changeicon"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Turn off"
/>
<TextView
android:id="@+id/text_hoverarea"
android:layout_width="match_parent"
android:layout_height="100dip"
android:layout_gravity="center_horizontal"
android:background="#FFDDDDDD"
android:text="Please hover the pen"
/>
</LinearLayout>
```

```
packagecom.example.pointericon;

importandroid.app.Activity;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.Button;
importandroid.widget.TextView;

importcom.samsung.android.sdk.look.SlookPointerIcon;

publicclassMainActivityextends Activity {

privatebooleanmIsDefault = false;
```

```
privateTextViewmHoverTextView;

privateSlookPointerIconmPointerIcon = newSlookPointerIcon();

@Override
publicvoidonCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main);

        Button changeIcon = (Button) findViewById(R.id.btn_changeicon);

mHoverTextView = (TextView) findViewById(R.id.text_hoverarea);
mPointerIcon.setHoverIcon(mHoverTextView, getResources()
                .getDrawable(R.drawable.ic_launcher));
changeIcon.setOnClickListener(newView.OnClickListener() {

                    publicvoidonClick(View v) {
                          if (mIsDefault) {
                                mPointerIcon.setHoverIcon(mHoverTextView,
getResources()
                                                .getDrawable(R.drawable.ic_launcher));
                                mIsDefault = false;
                                ((Button) v).setText("Turn off");
                          } else {
                                mPointerIcon.setHoverIcon(mHoverTextView, null);
                                mIsDefault = true;
                                ((Button) v).setText("Turn on");
                          }
                    }
              });
      }
}
```

3.2. Using PointerIcon

You can use `setHoverIcon()` to set or unset the pointer image when the pointer enters a drawable class on View.

3.2.1. Setting theHoveringPointer

To set a hovering pointer:

1. Create anSlookPointerIcon instance.
2. Set an icon image to the View instance with `setHoverIcon()`. If the pointer hovers over a View instance, the pointer changes to the image.

```
publicclassMainActivityextends Activity {

      /* Create PointerIcon instance */
      privateSlookPointerIconmPointerIcon = newSlookPointerIcon();
```

```
        @Override
        protectedvoidonCreate(Bundle savedInstanceState) {
<skip >

/* Set the icon */
              mPointerIcon.setHoverIcon(mHoverTextView, getResources()
                 .getDrawable(R.drawable.ic_launcher));

        }

}
```

3.2.2. UnsettingtheHovering Pointer

To unset a hovering pointer:

Call setHoverIcon() and pass null as the second parameter to unregister the current hovering pointer.

```
/* Unset the icon */
mPointerIcon.setHoverIcon(mHoverTextView, null);
```

4. AirButton

AirButton is a quick access menu via S-Pen to the recently used menus or content. You can use AirButton to allow users to click anAirButtonmenu, insert an image at the point and much more. You can use AirButton to provide quick access menu for various contexts.

Users can open and use AirButtonwithS-Pen. You can customize the gravity factor, the direction factor and the display type for AirButton in your application.

4.1. Hello AirButton

Hello AirButton is a sample application that:

- Creates an AirButton instance
- Creates a text menu as a sub-menu

Add the following layout to your activity_main.xml file for the sample application.

```
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<Button
android:id="@+id/button1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="100dip"
android:text="AirButton"/>

</LinearLayout>
```

```
packagecom.example.airbutton;

importjava.util.ArrayList;

importandroid.app.Activity;
importandroid.content.Context;
importandroid.os.Bundle;
importandroid.view.Menu;
importandroid.view.View;
importandroid.widget.Button;
importandroid.widget.Toast;

importcom.samsung.android.sdk.look.airbutton.SlookAirButton;
import com.samsung.android.sdk.look.airbutton.SlookAirButton.ItemSelectListener;
importcom.samsung.android.sdk.look.airbutton.SlookAirButtonAdapter;
import com.samsung.android.sdk.look.airbutton.SlookAirButtonAdapter.AirButtonItem;

publicclassMainActivityextends Activity {
```

```
        private Context vContext = null;
        private Button vBtnText = null;

        @Override
        protectedvoidonCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                vContext = this;

                setContentView(R.layout.activity_main);
                vBtnText = (Button) findViewById(R.id.button1);
                createTextListWidgetFromView(vBtnText);
        }

        privateItemSelectListenervCallback = newItemSelectListener() {
                @Override
                publicvoidonItemSelected(View v, intitemIndex, Object data) {
                        Toast.makeText(vContext, "Item index = " + itemIndex,
                                        Toast.LENGTH_SHORT).show();
                }
        };

        publicSlookAirButtoncreateTextListWidgetFromView(View v) {
                SlookAirButtonairButtonWidget = newSlookAirButton(v,
                                getAdapterStringList(), SlookAirButton.UI_TYPE_LIST);
                airButtonWidget.setItemSelectListener(vCallback);
                airButtonWidget.setPosition(0, 0);

                returnairButtonWidget;
        }

        publicSlookAirButtonAdaptergetAdapterStringList() {
                ArrayList<AirButtonItem>stringList = newArrayList<AirButtonItem>();
                stringList.add(newAirButtonItem(null, "1st Text Menu", null));
                stringList.add(newAirButtonItem(null, "2nd Text Menu", null));
                stringList.add(newAirButtonItem(null, "3rd Text Menu", null));

                returnnewSlookAirButtonAdapter(stringList);
        }
}
```

4.2. Using AirButton

Users can open an optimized menu for various contexts by pressing the side button on S-Pen.

4.2.1. Creating a Text AirButton

To create a text menu as a sub-menu for AirButton:

1. Create an AirButton instance with the type set to UI_TYPE_LIST.
2. Set your ItemSelectListener instance.
3. Set the position of the Air Button widget.
4. Create an adapter for the text string.

5. Add the menu texts.

6. Create a text menu as a sub-menu.

```
publicSlookAirButtoncreateTextListWidgetFromView(View v) {
    /* Create AirButton instance with UI_TYPE_LIST */
SlookAirButtonairButtonWidget    =    newSlookAirButton(v,    getAdapterStringList(),
SlookAirButton.UI_TYPE_LIST);
    /* Set item select listener */
airButtonWidget.setItemSelectListener(mCallback);
    /* Set position */
airButtonWidget.setPosition(0, 50);

returnairButtonWidget;
}

/* Create adapter for text string */
publicSlookAirButtonAdaptergetAdapterStringList() {
ArrayList<AirButtonItem>stringList = newArrayList<AirButtonItem>();
stringList.add(newAirButtonItem(null, "/* text string */", null));
    …
returnnewSlookAirButtonAdapter(stringList);
}
```

4.2.2. Creating an Image AirButton

To create an image menu as a sub-menu for AirButton:

1. CreateanAirButton instance with the type set to UI_TYPE_LIST.

2. Set your ItemSelectListener instance.

3. Set the gravity for the AirButton instance.

4. Set the direction for the AirButtoninstance.List items with DIRECTION_LEFT or DIRECTION_RIGHT are not supported.

5. Set the position for the AirButton instance.

6. Create an adapter for the image.

7. Set the image.

```
publicSlookAirButtoncreateImageListWidgetFromView(View v) {
    /* Create AirButton instance with UI_TYPE_LIST */
SlookAirButtonairButtonWidget = newSlookAirButton(v, getAdapterImageList(),
SlookAirButton.UI_TYPE_LIST);
    /* Set item select listener */
airButtonWidget.setItemSelectListener(mCallback);
    /* Set gravity */
airButtonWidget.setGravity(SlookAirButton.GRAVITY_LEFT);
    /* Set direction */
```

```
airButtonWidget.setDirection(SlookAirButton.DIRECTION_UPPER);
    /* Set position */
airButtonWidget.setPosition(0, -50);

returnairButtonWidget;
}

/* Create adapter for image */
publicSlookAirButtonAdaptergetAdapterImageList() {
ArrayList<AirButtonItem>itemList = newArrayList<AirButtonItem>();
itemList.add(newAirButtonItem(getResources().getDrawable(/* image */), null, null));
    …

returnnewSlookAirButtonAdapter(itemList);
}
```

4.2.3. Creating a Recipient List AirButton

To create a recipient list as a sub-menu for AirButton:

1. Create an AirButton instance with the type set to UI_TYPE_LIST.
2. Set the direction for the AirButtoninstance.List items with DIRECTION_LEFT or DIRECTION_RIGHT are not supported.
3. Set your ItemSelectListener instance.
4. Create an adapter for your recipient list.
5. Set the image for the menu item.
6. Set the text for the menu item.

```
publicSlookAirButtoncreateRecipientListWidgetFromView(View v) {
    /* Create AirButton instance with UI_TYPE_LIST */
SlookAirButtonairButtonWidget = newSlookAirButton(v, getAdapterRecipientList(),
SlookAirButton.UI_TYPE_LIST);
    /* Set direction */
airButtonWidget.setDirection(SlookAirButton.DIRECTION_LOWER);
    /* Set item select listener */
airButtonWidget.setItemSelectListener(mCallback);

returnairButtonWidget;
}

/* Create adapter for recipient list */
publicSlookAirButtonAdaptergetAdapterRecipientList() {
ArrayList<AirButtonItem>itemList = newArrayList<AirButtonItem>();
itemList.add(newAirButtonItem(getResources().getDrawable(R.drawable.recipient),
"Alexander Hamilton", null));
itemList.add(newAirButtonItem(/* image */, /* text */, null));
    …

returnnewSlookAirButtonAdapter(itemList);
}
```

4.2.4. Creating a Menu List AirButton

To create a legacy menu list as a sub-menu for AirButton:

1. Create an AirButton instance with the type set to UI_TYPE_MENU.
2. Set the direction for the AirButtoninstance.List items with DIRECTION_LEFT or DIRECTION_RIGHT are not supported.
3. Create an adapter for your menu list.

```
Publicclass MainActivity extends Activity {
      public SlookAirButton createMenuWidgetFromView(View v) {
   /* Create AirButton instance using UI_TYPE_MENU. */
            SlookAirButton airButtonWidget = new SlookAirButton(v,
                        getAdapterMenuList(), SlookAirButton.UI_TYPE_MENU);
   /* Set direction */
            airButtonWidget.setDirection(SlookAirButton.DIRECTION_RIGHT);

            return airButtonWidget;
      }

        /* Create adapter for menu list*/
      public SlookAirButtonAdapter getAdapterMenuList() {
            ArrayList<AirButtonItem> itemList = new ArrayList<AirButtonItem>();
            itemList.add(new AirButtonItem(mContext.getResources().getDrawable(
                        R.drawable.ic_menu_add), "Add", null));
            itemList.add(new AirButtonItem(mContext.getResources().getDrawable(
                        R.drawable.ic_menu_archive), "Help", null));
            itemList.add(new AirButtonItem(mContext.getResources().getDrawable(
                        R.drawable.ic_menu_edit), "Edit", null));
            itemList.add(new AirButtonItem(mContext.getResources().getDrawable(
                        R.drawable.ic_menu_help), "Help", null));
            ;

            return new SlookAirButtonAdapter(itemList);
      }
```

4.2.5. Using the DefaultAdapters

4.2.5.1 Using SlookAirButtonFrequentContactAdapter

The Look package offersyou a default adapter, SlookAirButtonFrequentContactAdapter, to configure AirButton to display frequently accessed contacts.

To use the default adapter, add the following permission in your Android manifest file.

```
<manifest>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
<application>
```

The following sample code shows how to use SlookAirButtonFrequentContactAdapter.

The arg2 parameter is a bundle type value that contains the display name, data, URI, and thumbnail image.

```
publicSlookAirButtoncreateRecipientListWidgetFromView(View v) {
        Bundle option = new Bundle();
option.putString("MIME_TYPE", "vnd.android.cursor.item/phone_v2");
SlookAirButtonairButtonWidget = newSlookAirButton(v,
newSlookAirButtonFrequentContactAdapter(v, null), SlookAirButton.UI_TYPE_LIST);
airButtonWidget.setDirection(SlookAirButton.DIRECTION_UPPER);
airButtonWidget.setItemSelectListener(newItemSelectListener() {

publicvoidonItemSelected(View arg0, int arg1, Object arg2) {

                Bundle bundle = (Bundle) arg2;
                String                              name                                =
bundle.getString(SlookAirButtonFrequentContactAdapter.DISPLAY_NAME);
                String                              data                                =
bundle.getString(SlookAirButtonFrequentContactAdapter.DATA);

Toast.makeText(AirButtonDefaultActivity.this, name + ":" + data, Toast.LENGTH_SHORT)
                        .show();
            }
        });

returnairButtonWidget;
}
```

4.2.5.2 Using SlookAirButtonRecentMediaAdatper

The Look package offers you a default adapter, SlookAirButtonRecentMediaAdapter, to configure AirButton to display recently played media files.

The following sample code shows how to use SlookAirButtonRecentMediaAdapter:

- The second parameter for the adapter is a bundle type value that contains the Boolean value true or false for IMAGE_TYPE, VIDEO_TYPE, and AUDIO_TYPE, for example,Bundle.setBoolean(IMAGE_TYPE, true).
- Cast the arg2 parameter to a URI.

Only UI_TYPE is available for this default adapter, and it is used as the parameter for the SlookAirButton constructor.

```
publicSlookAirButtoncreateImageListWidgetFromView(View v) {

SlookAirButtonairButtonWidget = newSlookAirButton(v,
newSlookAirButtonRecentMediaAdapter(v, null), SlookAirButton.UI_TYPE_LIST);
airButtonWidget.setItemSelectListener(newItemSelectListener() {

publicvoidonItemSelected(View arg0, int arg1, Object arg2) {

                Uri uri = (Uri) arg2;
```

```
                Toast.makeText(AirButtonDefaultActivity.this,            uri.toString(),
Toast.LENGTH_SHORT)
                        .show();
            }
        });
airButtonWidget.setGravity(SlookAirButton.GRAVITY_LEFT);
airButtonWidget.setDirection(SlookAirButton.DIRECTION_UPPER);
airButtonWidget.setPosition(0, -50);

returnairButtonWidget;
}
```

5. WritingBuddy

WritingBuddy is a popup panel for hand written input using S-Pen. When S-Pen approaches the user input area on the viewport, the writing panel pops up, and userscan write text with S-Pen.

Userscan input text more quickly with WritingBuddythanwith a soft keyboard.

Userscan use WritingBuddy for writing and editing text, and also for other menus and options such as dialer and date pickers.

WritingBuddy can recognize hand-written text, numbers and hand-drawn images.

5.1. Hello WritingBuddy

Hello WritingBuddy is a sample application that:

- Creates a WritingBuddy instance
- Handles the input

Add the following layout to your activity_main.xml file.

```
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/information"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal">

<Button
android:id="@+id/btn_enable"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Disable"/>

<Button
android:id="@+id/btn_type"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Number"/>
</LinearLayout>

<FrameLayout
android:id="@+id/input"
android:layout_width="match_parent"
android:layout_height="200dip"
android:background="#FFDDDDDD">
```

```
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:gravity="center"
android:text="Please write here by S-Pen"/>
</FrameLayout>

<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="[OUTPUT]"/>

<TextView
android:id="@+id/text_output"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:gravity="left"
android:text=""
android:textColor="#FF0000FF"/>

</LinearLayout>
```

```
publicclassWritingBuddyViewGroupActivityextends Activity {

	privateSlookWritingBuddymWritingBuddy;

	privateTextViewmOutputTextView;

	/* State */
	privatebooleanmIsEnabled = true;

	privateintmType = SlookWritingBuddy.TYPE_EDITOR_TEXT;

	@Override
	publicvoidonCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_writingbuddy_viewgroup);

		FrameLayoutfl = (FrameLayout) findViewById(R.id.input);
		mOutputTextView = (TextView) findViewById(R.id.text_output);

		mWritingBuddy = newSlookWritingBuddy(fl);

		mWritingBuddy

	.setTextWritingListener(newSlookWritingBuddy.TextWritingListener() {

				publicvoidonTextReceived(CharSequence arg0) {
					mOutputTextView.setText(arg0);
				}
			});

		Button enableButton = (Button) findViewById(R.id.btn_enable);
		enableButton.setOnClickListener(newView.OnClickListener() {

			publicvoidonClick(View v) {
```

```
                        if (mIsEnabled) {
                                ((Button) v).setText("Enable");
                                mIsEnabled = false;
                        } else {
                                ((Button) v).setText("Disable");
                                mIsEnabled = true;
                        }
                        mWritingBuddy.setEnabled(mIsEnabled);
                }
        });

        Button typeButton = (Button) findViewById(R.id.btn_type);
        typeButton.setOnClickListener(newView.OnClickListener() {

                publicvoidonClick(View v) {
                        if (mType == SlookWritingBuddy.TYPE_EDITOR_TEXT) {
                                mType = SlookWritingBuddy.TYPE_EDITOR_NUMBER;
                                ((Button) v).setText("String+Number");
                        } else {
                                mType = SlookWritingBuddy.TYPE_EDITOR_TEXT;
                                ((Button) v).setText("Number");
                        }
                        mWritingBuddy.setEditorType(mType);
                }
        });
    }
}
```

5.2. Using WritingBuddy

WritingBuddy recognizes hand-written text or hand-drawn images and sends the data to View.

You can use the following methods to use WritingBuddy in your applications:

- `SlookWritingBuddy.getEditorType()`gets the editor type used in the input panel.
- `SlookWritingBuddy.isEnabled()`checksifWritingBuddyis enabled
- `SlookWritingBuddy.setEditorType()` sets the editor type used in the input panel.
- `SlookWritingBuddy.setEnabled()` enables or disablesWritingBuddy.
- `SlookWritingBuddy.setImageWritingListener()`sets the listener to be called when an image is committed from the input panel.
- `SlookWritingBuddy.setTextWritingListener()` sets the listener to be called when text is written from the input panel.
- `SlookWritingBuddy.ImageWritingListener.onImageReceived()`called when hand-written input is completed if image capturing is enabled.
- `SlookWritingBuddy.TextWritingListener.onTextReceived()`called when hand-written input is completed if image capturing is not enabled.

5.2.1. Receiving Hand-Written Text Input

To receive hand-written text input with S-Pen:

1. Create anSlookWritingBuddy instance.
2. Set your TextWritingListener instance for ViewGroup.
3. Implement the listener method.

```
FrameLayoutfl = (FrameLayout) findViewById(R.id.input);
mOutputTextView = (TextView) findViewById(R.id.text_output);
/* Create SlookWritingBuddy instance */
mWritingBuddy = newSlookWritingBuddy(fl);
/* In case of ViewGroup, set textwriting listener */
mWritingBuddy.setTextWritingListener(newSlookWritingBuddy.TextWritingListener() {

publicvoidonTextReceived(CharSequence arg0) {
        /* Implement listener method */
mOutputTextView.setText(arg0);
    }
});
```

5.2.2. Receiving Hand-Written Image Input

Use EditTexttoreceive hand-written image input. The input should be drawn with an S-Pen.

To receive hand-written image input with S-Pen:

1. Create anSlookWritingBuddy instance.
2. Set your ImageWritingListener instance for EditText.
3. Implement the listener method.

```
privateSlookWritingBuddymWritingBuddy;
…
/* Create SlookWritingBuddy instance */
mWritingBuddy = newSlookWritingBuddy(editInput);
…
/* In case of Edittext, set ImageWritingListiner for image writing */
mWritingBuddy.setImageWritingListener(newSlookWritingBuddy.ImageWritingListener() {
publicvoidonImageReceived(Bitmap arg0) {
        /* Implement listener method */
mOutputImageView.setBackground(newBitmapDrawable(getResources(), arg0));
    }
});
```

6. SmartClip

SmartClip is like an advanced screen shot capture technique. If you delineate an area on the screen, SmartClip collects and stores metadata such as text and URL information from the enclosed area. Users can use SmartClip with S-Pen when in Easy clip or Scrapbooker mode.

By default, only the text and images are extracted from the captured area. If your application uses CustomView to create its pages, the internal data must be added. To provide additional information for the captured area within your application, use the Look package methods. For example, add a URL or deep links to SmartClip in your application for users to access them more easily.

You can add metatags directly to View or add metatags by registering a callback that runs when a screen shot is captured.

6.1. Hello SmartClip

Hello SmartClip is a sample application that:

- Creates a SmartClip instance
- Adds metatags for SmartClip

Add the following layout to your activity_main.xml file.

```
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:padding="50dip">

<com.samsung.android.example.slookdemos.CustomTextView
android:id="@+id/text_dynamic"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginBottom="100dip"
android:background="#FFEEEEEE"
android:text="Please draw closed circle by pen with pressing button."/>

<TextView
android:id="@+id/text_static"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:background="#FFDDDDDD"
android:text="Please draw closed circle by pen with pressing button."/>

<Button
android:id="@+id/gotopinboard"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="100dip"
```

```
android:text="Go to Scrapbook"/>

</LinearLayout>
```

```
packagecom.samsung.android.example.slookdemos;

importcom.samsung.android.sdk.look.smartclip.SlookSmartClip;
importcom.samsung.android.sdk.look.smartclip.SlookSmartClipMetaTag;

importandroid.app.Activity;
importandroid.content.ActivityNotFoundException;
importandroid.content.ComponentName;
importandroid.content.Intent;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.Button;
importandroid.widget.TextView;
importandroid.widget.Toast;

publicclassSmartClipActivityextends Activity {

	@Override
	publicvoidonCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		setContentView(R.layout.activity_smartclip);

		TextViewtv = (TextView) findViewById(R.id.text_static);

		SlookSmartClipsc = newSlookSmartClip(tv);
		sc.clearAllMetaTag();
		sc.addMetaTag(newSlookSmartClipMetaTag(
				SlookSmartClipMetaTag.TAG_TYPE_URL,
"http://www.samsung.com"));
		sc.addMetaTag(newSlookSmartClipMetaTag(
				SlookSmartClipMetaTag.TAG_TYPE_PLAIN_TEXT,
				"This is android textview."));

		Button gotoPinAll = (Button) findViewById(R.id.gotopinboard);
		gotoPinAll.setOnClickListener(newView.OnClickListener() {

			publicvoidonClick(View v) {
				Intent intent = new Intent();
				intent.setComponent(newComponentName(
						"com.samsung.android.app.pinboard",

	"com.samsung.android.app.pinboard.ui.PinboardActivity"));
				try {
					startActivity(intent);
				} catch (ActivityNotFoundException e) {
					Toast.makeText(SmartClipActivity.this,
							"ScrapBook application is not
installed.",
							Toast.LENGTH_SHORT).show();
				}
			}
		});
```

```
        }
}
```

6.2. Using SmartClip

To extract better arranged and more useful data from the viewport, create a CustomTextView and collect the data withmetatags.

The Look package offers you the following methods for SmartClip in your application:

- `SlookSmartClip.addMetaTag()` adds metatag to the view.
- `SlookSmartClip.clearAllMetaTag()` removes all the metatagsof the view.
- `SlookSmartClip.extractDefaultSmartClipData()` extracts default metadatafromthe view.
- `SlookSmartClip.removeMetaTag()` removes the metadataofa tag type from the view.
- `SlookSmartClip.setDataExtractionListener()` sets a DataExtractionListener instance.
- `SlookSmartClip.onExtractSmartClipData()` called when SmartClipextractsSmartClipData.
- `SlookSmartClipCroppedArea.getRect()` gets the area of SmartClip.
- `SlookSmartClipCroppedArea.intersects()` determines whether a cropped area intersects with the given rect or view.
- `SlookSmartClipDataElement.addTag()` adds the metatagtothe element.
- `SlookSmartClipDataElement.getAllTag()` gets all metatags.
- `SlookSmartClipDataElement.getMetaAreaRect()` gets the area of the metadata.
- `SlookSmartClipDataElement.getTag()`getsthemetatagset with the given tag type.
- `SlookSmartClipDataElement.removeTag()` removes themetatagwith the specified tag type.
- `SlookSmartClipDataElement.setMetaAreaRect()` sets the metadataarea.
- `SlookSmartClipMetaTag.getType()` gets the metatagtype.
- `SlookSmartClipMetaTag.getValue()` gets the metatagvalue.
- `SlookSmartClipMetaTag.setType()` sets the metatagtype.
- `SlookSmartClipMetaTag.setValue()` sets the metatagvalue.
- `SlookSmartClipMetaTagArray.getTag()` gets themetatagwith the same tag type as this array
- `SlookSmartClipMetaTagArray.removeTag()` removes the metatagfrom this array.

6.2.1. Using SmartClip for Advanced Extraction

To add more information to the S-Pen-captured area, create CustomTextView for SmartClip.

To implement advanced extraction:

1. Create anSlookSmartClip instance.

2. Set your DataExtractionListener instance for SmartClip in the CustomTextView.
3. Implement the listener method
4. Extract the default metadata from the view. The first parameter is a variable to store extracted data and the second parameter represents the cropped area.
5. Add a metatag to the result element object.
6. Return an integer value to indicate whether the extracted data is used or discarded.

```
voidinit() {
/* Create SlookSmartClip instance */
    vSmartClip = newSlookSmartClip(this);

/* Set a DataExtractionListener of SmartClip at CustomtextView */
    vSmartClip.setDataExtractionListener(newSlookSmartClip.DataExtractionListener() {

        @Override
        publicintonExtractSmartClipData(View view,
       SlookSmartClipDataElementresultElement,
       SlookSmartClipCroppedAreaarg) {
       /* Extract default metadata from View. (First parameter is variable to store
extracted data, second parameter represents cropped area) */
       vSmartClip.extractDefaultSmartClipData(resultElement,arg);

/* Add a metatag to the resultElement object */
       SlookSmartClipMetaTagmetaTag                                                =
newSlookSmartClipMetaTag(SlookSmartClipMetaTag.TAG_TYPE_URL,
“http://www.samsung.com”);

       resultElement.addTag(metaTag);

/* Return value indicating whether the extracted data was used or discarded */
       returnSlookSmartClip.DataExtractionListener.EXTRACTION_DEFAULT;
         }
    });
}
```

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/