SAMSUNG

SAMSUNG

Samsung Programmable Key for Non-Rugged Devices

Get more done. At the Press of a Button.

Supported Devices

- Non Rugged Devices Running OS 12 & above currently
- Non Rugged Devices that receive the OS 12 maintenance release in the future.
- Examples:

Flagship	A Series	Tablet
Galaxy S22 Series	Galaxy A53 Series	Tab S8 Series
Galaxy S21 Series	Galaxy A52 Series	Tab S7 Series
Galaxy S20 Series	Galaxy A33 Series	
Galaxy S10 Series	Galaxy A32 Series	
Galaxy Note 20 Series		
Galaxy Note 10 Series		
…and more to come		

Partner Journey

1. Choose Programmable Intent Model
2. Make App Modifications (Optional depending on chosen model)
3. Release Application

Prerequisites

Supported Devices:

- Devices must be on a minimum of Knox 3.8/Android 12 (S OS).
- Only the Power button can be remapped at this time.

Licenses:

- A Knox License Key (free) is required to use this functionality, but license activation is not required to be built into partners' app.
- Knox Service Plugin (KSP) will handle the license activation on behalf of partners.
- For Partner testing, set up a KSP policy via a UEM (all major UEMs are compatible) to test. Follow IT admin instructions.
- Partners can retrieve Knox License keys through KPP, our Knox Partner program (link below), or request from a Samsung Rep.

Rugged Device Intents:

- The intents in this document do not replace the intents available on Rugged devices, but instead work in tandem.

UEM - KSP (Knox Service Plugin):

- UEM and Android Enterprise is required to use KSP.
- For devices where Bixby is the default mapping, an EMM policy disabling Bixby using the Bixby package names is required.
- Screen on/off interference will vary per device when the Home Key is pressed. In general, long press is recommended to initiate PTT on Non-Rugged devices.

More info on KSP: https://docs.samsungknox.com/admin/knox-service-plugin/welcome.htm
More info on KPP: https://partner.samsungknox.com/

Intent Model

Listening to the side key intent can be done in 2 ways based on the IT admin configuration:

1. Partner Application can Listen for Samsung intent(s) on PRESS and RELEASE
2. Partner Application can Listen for its own intents. Samsung will proxy PRESS and RELEASE events to that intent.

Listening for Samsung Intent

Step 1: Define a broadcast receiver :

```
<receiver
android:name=".KeymappingReceiver"
android:enabled="true"
android:exported="true">
<intent-filter>
<action android:name="com.samsung.android.knox.intent.action.HARD_KEY_REPORT" />
</intent-filter>
</receiver>
```

Listening for Samsung Intent (continued)

Step 2 : Define the broadcast receiver java class

```
public class KeymappingReceiver extends BroadcastReceiver {

private static final String TAG = "KeymappingReceiver";

private static final String ACTION_SAMSUNG_INTENT = "com.samsung.android.knox.intent.action.HARD_KEY_REPORT";
private static final String EXTRA_KEY_CODE = "com.samsung.android.knox.intent.extra.KEY_CODE";
private static final String EXTRA_REPORT_TYPE_NEW = "com.samsung.android.knox.intent.extra.KEY_REPORT_TYPE_NEW";

private static final int KEYCODE_SIDE = 26;
private static final int KEY_PRESS = 1;
private static final int KEY_RELEASE = 2;

@Override
public void onReceive(Context context, Intent intent) {
Log.i(TAG, "@onReceive - " + intent.getAction());

if (intent != null & ACTION_SAMSUNG_INTENT.equalsIgnoreCase(intent.getAction())) {
// Samsung intent
int keyCode = intent.getIntExtra(EXTRA_KEY_CODE, 0); // 26 for side key
int keyPressType = intent.getIntExtra(EXTRA_REPORT_TYPE_NEW, -1); // 1 for key press; 2 for key release
Log.i(TAG, "keyCode == " + keyCode + " keyPressType == " + keyPressType);

//TODO handle below...
}
}//onReceive
}
```

Listening for Custom(Partner) Intents

For Partners using custom intents (Partner defined intents), please ensure the manifest of the Partner application meets the example guidelines on the following pages.

When custom intents are defined in the Knox Policy, the Knox Client (KSP) will proxy all press and release button events to the Partner's intent.

The Partner application may already be compliant, and as such, app modifications may not be required.

Note: For Partners who intend to use custom intents, documentation of those intents will need to be supplied to your customers for use in the Knox Policy that customers will need to create in their EMM to configure the buttons.

Listening for Custom Intents (Sample Code)

Step 1: Define a broadcast receiver

```
<receiver
android:name=".KeymappingReceiver"
android:enabled="true"
android:exported="true">
<intent-filter>
<action android:name="com.samsung.android.knox.intent.action.SIDE_PRESS" />
<action android:name="com.samsung.android.knox.intent.action.SIDE_RELEASE" />
</intent-filter>
</receiver>
```

Listening for Custom Intents (continued)

Step 2: Define the broadcast receiver java class

```
public class KeymappingReceiver extends BroadcastReceiver {

    private static final String TAG = "KeymappingReceiver";

    private static final String EXTRA_KEY_PRESS_TIMESTAMP = "com.samsung.android.knox.extra.EVENT_TIMESTAMP";
    private static final String ACTION_SIDE_KEY_PRESS_DOWN = "com.samsung.android.knox.intent.action.SIDE_PRESS";
    private static final String ACTION_SIDE_KEY_PRESS_UP = "com.samsung.android.knox.intent.action.SIDE_RELEASE";

    @Override
    public void onReceive(Context context, Intent intent) {
        Log.i(TAG, "@onReceive - " + intent.getAction());

        if (intent != null && intent.getAction() != null) {

        //For custom intent, need to use intent action to distinguish key and action type
        long timeStamp = intent.getLongExtra(EXTRA_KEY_PRESS_TIMESTAMP, -1L);

        if (ACTION_SIDE_KEY_PRESS_DOWN.equalsIgnoreCase(intent.getAction())) {
            // Side key is pressed, your code here
            Log.i(TAG, “Side key pressed");
         } else if (ACTION_SIDE_KEY_PRESS_UP.equalsIgnoreCase(intent.getAction())) {
            // Side key is released, your code here
           Log.i(TAG, “Side key released");
         }
      }
    }//onReceive
}
```