
	Doc.Type API Guide	Author SW R&D Dept.		Page of Pages 1 / 9
	Model PM Android Models	Doc.No.	Rev. R02	Date 2020.01.30

Emkit Intent API Guide for SDK v3

Pointmobile Co., Ltd.
S/W R&D Team

COMPANY CONFIDENTIAL
Only to be disclosed to Point Mobile employees concerned with the project.
DO NOT MAKE COPIES

	Doc.Type API Guide	Author SW R&D Dept.		Page of Pages 2 / 9
	Model PM Android Models	Doc.No.	Rev. R02	Date 2020.01.30

Emkit Intent API Guide for SDK v3

Revision History

Revision	Date	Change Description	Author
R01	2020.01.20	Initial draft	Ryan Kim
R02	2020.01.30	Translate to English	Emma Jung



	Doc.Type API Guide	Author SW R&D Dept.		Page of Pages 3 / 9
	Model PM Android Models	Doc.No.	Rev. R02	Date 2020.01.30

Table of Contents

1. Broadcast Intents	4
device.common.ENABLED_SCANNER.....	4
device.common.ENABLED_TRIGGER	4
device.common.ENABLED_TOUCHSCAN.....	5
device.common.READ_TOUCHSCAN_STATE	5
 2. onReceive Intents.....	 6
device.common.USERMSG	6
device.scanner.EVENT	7
device.common.STATE_TOUCHSCAN.....	8
 3. AIM Identifier	 9

	Doc.Type API Guide	Author SW R&D Dept.		Page of Pages 4 / 9
	Model PM Android Models	Doc.No.	Rev. R02	Date 2020.01.30

1. Broadcast Intents

You can set or read the scanner status values with the following intents from the user application.

device.common.ENABLED_SCANNER

Used to enable/disable the scanner.

The default status is 'Disable' and changed status will be retained after the device restart.

(* Note: In Setting App - Settings > ScanSetting > Enable/Disable *)

Action	"device.common.ENABLED_SCANNER"
Parameter	Extra: "EXTRA_ENABLED_SCANNER" (int) 0 – Disabled 1 – Enabled
Example	<p>How to enable the scanner :</p> <pre>Intent scannerEnableIntent = new Intent("device.common.ENABLED_SCANNER"); scannerEnableIntent.putExtra("EXTRA_ENABLED_SCANNER", 1); sendBroadcast(scannerEnableIntent);</pre> <p>How to disable the scanner :</p> <pre>Intent scannerDisableIntent = new Intent("device.common.ENABLED_SCANNER"); scannerDisableIntent.putExtra("EXTRA_ENABLED_SCANNER", 0); sendBroadcast(scannerDisableIntent);</pre>

device.common.ENABLED_TRIGGER


Used to enable/disable the trigger action(starting action to read the barcode) of the enabled scanner.

Changing the trigger status is usually used to disable reading the barcode at a specific moment and it takes less time than disabling the entire scanner.

The status value will be reset to 'Enable(Default Status)' after the device restart.

(* Note: In Setting App – Trigger action status cannot be set in ScanSetting, only can by API or Intent *)

Action	"device.common.ENABLED_TRIGGER"
Parameter	Extra: "EXTRA_ENABLED_TRIGGER" (int) 0 – Disabled 1 – Enabled
Example	<p>How to enable the trigger :</p> <pre>Intent triggerEnableIntent = new Intent("device.common.ENABLED_TRIGGER"); triggerEnableIntent.putExtra("EXTRA_ENABLED_TRIGGER", 1); sendBroadcast(triggerEnableIntent);</pre> <p>How to disable the trigger :</p> <pre>Intent triggerDisableIntent = new Intent("device.common.ENABLED_TRIGGER"); triggerDisableIntent.putExtra("EXTRA_ENABLED_TRIGGER", 0); sendBroadcast(triggerDisableIntent);</pre>

	Doc.Type API Guide	Author SW R&D Dept.		Page of Pages 5 / 9
	Model PM Android Models	Doc.No.	Rev. R02	Date 2020.01.30

device.common.ENABLED_TOUCHSCAN

Used to enable/disable the TouchScan button, which is a trigger button on the device's screen.

The button can be shown and activate only when the scanner is enabled.

And even if the scanner is enabled, the TouchScan button is not shown when you enter image mode.

The button default status is 'Disable' and changed status will be retained after the device restart.

(* Note : In Setting App - Settings > ScanSetting > Basic > Enable TouchScan *)

Action	"device.common.ENABLED_TOUCHSCAN"
Parameter	Extra: "EXTRA_ENABLED_TOUCHSCAN" (int) 0 – Disabled 1 – Enabled
Example	<p>How to enable the TouchScan :</p> <pre>Intent touchScanEnableIntent = new Intent("device.common.ENABLED_TOUCHSCAN"); touchScanEnableIntent.putExtra("EXTRA_ENABLED_TOUCHSCAN", 1); sendBroadcast(touchScanEnableIntent);</pre> <p>How to disable the TouchScan :</p> <pre>Intent touchScanDisableIntent = new Intent("device.common.ENABLED_TOUCHSCAN"); touchScanDisableIntent.putExtra("EXTRA_ENABLED_TOUCHSCAN", 0); sendBroadcast(touchScanDisableIntent);</pre>


device.common.READ_TOUCHSCAN_STATE

Used to read the enable/disable status of the TouchScan button.

After sending this intent, you can get the status value of the TouchScan button with

device.common.STATE_TOUCHSCAN intent.

Action	"device.common.READ_TOUCHSCAN_STATE"
Parameter	None
Example	<pre>Intent readTouchScanStateIntent = new Intent("device.common.READ_TOUCHSCAN_STATE"); sendBroadcast(readTouchScanStateIntent);</pre>

	Doc.Type API Guide	Author SW R&D Dept.		Page of Pages 6 / 9
	Model PM Android Models	Doc.No.	Rev. R02	Date 2020.01.30

2. onReceive Intents

You can get the scan result or scanner status value with the following intents from user application.

device.common.USERMSG

Used to get the success/fail result after decoding the barcode.


This intent will run only when the return value of `aDecodeGetResultType()` equals

'DCD_RESULT_USERMSG' (Result type in [ScanSetting > Basic > Wedge mode] is 'User Message').

The user application must have the receiver for this intent.

If you want to get a detailed result of decoding, use `aDecodeGetResult()` method.

Action	"device.common.USERMSG"
Parameter	Extra: "EXTRA_USERMSG" (boolean) false – Decoding fail true – Decoding success
Example	<pre> public class ScanResultReceiver extends BroadcastReceiver { @Override public void onReceive(Context context, Intent intent) { if (mScanner != null) { if ("device.common.USERMSG".equals(intent.getAction())) { boolean isReadSuccess = intent.getBooleanExtra("EXTRA_USERMSG", false); if (isReadSuccess) { mScanner.aDecodeGetResult(mDecodeResult.recycle()); Log.d(TAG, "SCAN SUCCESS : " + mDecodeResult.toString()); } else { Log.d(TAG, "SCAN FAIL !!"); } } } } } </pre>

	Doc.Type API Guide	Author SW R&D Dept.		Page of Pages 7 / 9
	Model PM Android Models	Doc.No.	Rev. R02	Date 2020.01.30

device.scanner.EVENT


Used to receive both the success/fail result and detail result data after decoding the barcode.

This intent will run only when the return value of `aDecodeGetResultType()` equals

'RESULT_EVENT' (Result type in [ScanSetting > Basic > Wedge mode] is 'Intent Broadcast').

The user application must have the receiver for this intent.

Action	"device.common.EVENT"
Parameters	<p>Extra: "EXTRA_EVENT_DECODE_RESULT" (boolean) false – Decoding fail true – Decoding success</p> <p>Extra: "EXTRA_EVENT_DECODE_LENGTH" (int) The length of decoding data (= the length of result of 'EXTRA_EVENT_DECODE_VALUE')</p> <p>Extra: "EXTRA_EVENT_DECODE_VALUE" (byte[]) The value of decoding data. The format is byte[] and byte[] is usually converted to String to use.</p> <p>Extra: "EXTRA_EVENT_DECODE_LETTER" (byte) The decoded barcode's code character in AIM identifier</p> <p>Extra: "EXTRA_EVENT_DECODE_MODIFIER" (byte) The decoded barcode's modifier character in AIM identifier</p> <p>Extra: "EXTRA_EVENT_DECODE_TIME" (int) Time from trigger start to complete the decoding (unit : ms)</p> <p>Extra: "EXTRA_EVENT_SYMBOL_NAME" (string) The name of decoded barcode</p> <p>Extra: "EXTRA_EVENT_SYMBOL_ID" (byte) The barcode ID of decoded barcode. (ScanSetting > Symbologies > (Symbology name) > Barcode ID)</p> <p>Extra: "EXTRA_EVENT_SYMBOL_TYPE" (int) The type of decoded barcode (= ScanConst.SymbologyID value)</p>

	Doc.Type API Guide	Author SW R&D Dept.		Page of Pages 8 / 9
	Model PM Android Models	Doc.No.	Rev. R02	Date 2020.01.30

Example	<pre> public class ScanResultReceiver extends BroadcastReceiver { @Override public void onReceive(Context context, Intent intent) { if (mScanner != null) { if ("device.scanner.EVENT".equals(intent.getAction())) { boolean result = intent.getBooleanExtra("EXTRA_EVENT_DECODE_RESULT", false); int decodeBytesLength = intent.getIntExtra("EXTRA_EVENT_DECODE_LENGTH", 0); byte[] decodeBytesValue = intent.getByteArrayExtra("EXTRA_EVENT_DECODE_VALUE"); String decodeValue = new String(decodeBytesValue, 0, decodeBytesLength); int decodeLength = decodeValue.length(); String symbolName = intent.getStringExtra("EXTRA_EVENT_SYMBOL_NAME"); byte symbolId = intent.getByteExtra("EXTRA_EVENT_SYMBOL_ID", (byte) 0); int symbolType = intent.getIntExtra("EXTRA_EVENT_SYMBOL_TYPE", 0); byte letter = intent.getByteExtra("EXTRA_EVENT_DECODE_LETTER", (byte) 0); byte modifier = intent.getByteExtra("EXTRA_EVENT_DECODE_MODIFIER", (byte) 0); int decodingTime = intent.getIntExtra("EXTRA_EVENT_DECODE_TIME", 0); Log.d(TAG, "1. result: " + result); Log.d(TAG, "2. bytes length: " + decodeBytesLength); Log.d(TAG, "3. bytes value: " + decodeBytesValue); Log.d(TAG, "4. decoding length: " + decodeLength); Log.d(TAG, "5. decoding value: " + decodeValue); Log.d(TAG, "6. symbol name: " + symbolName); Log.d(TAG, "7. symbol id: " + symbolId); Log.d(TAG, "8. symbol type: " + symbolType); Log.d(TAG, "9. decoding letter: " + letter); Log.d(TAG, "10. decoding modifier: " + modifier); Log.d(TAG, "11. decoding time: " + decodingTime); } } } } </pre>
---------	--


device.common.STATE_TOUCHSCAN

Used to get the enable/disable status of the TouchScan button.

You can get the status value only if the user sends the `device.common.READ_TOUCHSCAN_STATE` intent.

The user application must have the receiver for this intent.

Action	"device.commom.STATE_TOUCHSCAN"
Parameter	Extra: "EXTRA_TOUCHSCAN_STATE" (int) 0 – Disabled 1 – Enabled
Example	<pre> private class TouchScanReceiver extends BroadcastReceiver { @Override public void onReceive(Context context, Intent intent) { if ("device.commom.STATE_TOUCHSCAN".equals(intent.getAction())) { int touchScanEnable = intent.getIntExtra("EXTRA_TOUCHSCAN_STATE", 0); if(touchScanEnable == 1) { Log.d(TAG, "TouchScan is enabled"); } else { Log.d(TAG, "TouchScan is disabled"); } } } } </pre>

	Doc.Type API Guide	Author SW R&D Dept.		Page of Pages 9 / 9
	Model PM Android Models	Doc.No.	Rev. R02	Date 2020.01.30

3. AIM Identifier

The AIM Code ID is a 3 character ISO/IEC identifier (originally created by AIM) generated by the decoder of a scanner, and gives information about the symbology of the barcode which was scanned.

Format	1 <i>cm</i>
Where	1 = flag character (ASCII 93) <i>c</i> = code character (defined by AIM for that symbology) <i>m</i> = modifier character