Honeywell Android Data Collection Intent API

Summary

The Data Collection Intent API is available on Honeywell Android handheld computers. It provides access to the scanner through a simple API, without installing a Honeywell SDK.

The application sends an Intent to claim the scanner and configure scanning, and another Intent to release the scanner. An optional third Intent controls scanning.

The application implements an Intent receiver to receive barcode events from the scanner.

Relationship to other Scanning APIs

Honeywell Android handheld computers provide several alternatives to allow applications to use the scanner. In order of increasing programming sophistication:

- Scan Wedge
- Data Collection Intent API
- Honeywell Mobility SDK for Android

Developer's Implementation Checklist

These are the steps to connect your application with a scanner through the Data Collection Intent API. For an example, see Code Snippet: MainActivity.

Application Code

Android Manifest

<uses-permission android:name="com.honeywell.decode.permission.DECODE" />

Activity.onResume()

Create an Intent receiver for barcode data.

Send intent to claim the scanner.

Activity.onPause()

Send intent to release the scanner.

Additional Scanner Control

In the simplest implementation, the user controls the scanner. If, instead, you need your application to control scanning, add code to send the Control-scanner Intent.

Configuring the Scanner

When using the ACTION_CLAIM_SCANNER Intent, you have two options for configuring the details of the scanning. Your application may rely on a separately defined profile, or it may directly set scanning properties.

Use a Scanning Profile

A scanning profile contains a collection of settings for the scanner.

To create a profile on a device, go to Android settings > Scanning. When creating the profile, select a custom name. Your application will select the profile by this name.

When creating the ACTION_CLAIM_SCANNER Intent, set the EXTRA_PROFILE extra string value to the profile name you created.

Using EZConfig for provisioning, the scanning profiles are in DataCollectionService.xml.

Programmatically Specify Properties

When creating the ACTION_CLAIM_SCANNER Intent, set the EXTRA_PROPERTIES Intent extra Bundle. The Bundle contains all of the scanning properties your application needs. The property names and values are documented in the com.honeywell.aidc.BarcodeReader class.

If both EXTRA_PROFILE and EXTRA_PROPERTIES are used to define scanning properties, then the EXTRA_PROPERTIES takes precedence.

Interface Details

Claim-scanner Intent

An application sends this intent to setup the scanner and begin exclusively receiving barcode data.

The normal behavior claims the internal scanner. If an external scanner is connected, then the default behavior instead connects to an external scanner.

Action: "com.honeywell.aidc.action.ACTION_CLAIM_SCANNER"

Data URI: None

MIME Type: None

Extras

> "com.honeywell.aidc.extra.EXTRA_PROFILE" (string): Specifies the profile name. To use the Default profile in Scanning settings, set this to "DEFAULT". If no profile name is provided, the scanner will use scanner factory default settings.

"com.honeywell.aidc.extra.EXTRA_SCANNER" (string): Value is a string. Overrides the default behavior selecting the scanner to use.

"dcs.scanner.imager": Claims the internal scanner.

"dcs.scanner.ring": Claims the external ring scanner.

"com.honeywell.aidc.extra.EXTRA_PROPERTIES" (Bundle): Bundle of properties which override the properties of the selected profile. The key names and values correspond to the properties of the com.honeywell.aidc.BarcodeReader class.

Release-scanner Intent

An application sends this intent to setup the scanner and begin exclusively receiving barcode data.

Action: "com.honeywell.aidc.action.ACTION_RELEASE_SCANNER"

Data URI: None

MIME Type: None

Control-scanner Intent

An application sends this intent to start the scanner.

Action: "com.honeywell.aidc.action.ACTION_CONTROL_SCANNER"

Data URI: None

MIME Type: None

Extras

"com.honeywell.aidc.extra.EXTRA_SCAN" (boolean): Set to true to start or continue scanning. Set to false to stop scanning. Most scenarios only need this extra, however the scanner can be put into other states by adding from the following extras.

"com.honeywell.aidc.extra.EXTRA_AIM" (boolean): Specify whether to turn the scanner aimer on or off. This is optional; the default value is the value of EXTRA_SCAN.

"com.honeywell.aidc.extra.EXTRA_LIGHT" (boolean): Specify whether to turn the scanner illumination on or off. This is optional; the default value is the value of EXTRA_SCAN.

"com.honeywell.aidc.extra.EXTRA_DECODE" (boolean): Specify whether to turn the decoding operation on or off. This is optional; the default value is the value of EXTRA_SCAN

Barcode Read Event Intent

An application receives this intent after a successful barcode scan. The action is determined by the "DPR_DATA_INTENT_ACTION" property.

Action: "DPR_DATA_INTENT_ACTION" property value.
i.e. "com.honeywell.aidc.action.ACTION_BARCODE_READ_EVENT"

Data URI: None

MIME Type: None

Extras

"aimId" (string): the AIM symbology identifier

"charset" (string) : the charset used to encode the raw barcode data to the barcode data string

"codeId" (string) : the Honeywell symbology identifier. See *Table 1: Honeywell Symbology Identifier Chart*

"data" (string): the barcode data string

"dataBytes" (byte[]) : the raw barcode data

"timestamp" (string) : the timestamp of the barcode data

"version" (int) : 1

Table 1: Honeywell Symbology Identifier Chart

Symbology	**Code Id**
DOTCODE	'.'
CODE1	'1'
MERGED_COUPON	';'
CODE39_BASE32, CODE32, ITALIAN PHARMACODE, PARAF	'<'
LABELCODE_IV	'>'
LABELCODE_V	'<'
TRIOPTIC	'='
KOREA_POST	'?'
INFOMAIL	','
EAN13_ISBN	'`'
SWEEDISH_POST	'['
RM_MAILMARK	'|'
BRAZIL_POST	']'
AUS_POST	'A'
BRITISH_POST	'B'
CANADIAN_POST	'C'
EAN8	'D'
UPCE	'E'

Symbology	Code Id
BC412	'G'
HAN_XIN_CODE	'H'
GS1_128	'I'
JAPAN_POST	'J'
KIX_CODE	'K'
PLANET_CODE	'L'
USPS_4_STATE, INTELLIGENT_MAIL	'M'
UPU_4_STATE, ID_TAGS	'N'
OCR	'O'
POSTNET	'P'
HK25, CHINA_POST	'Q'
MICROPDF	'R'
SECURE_CODE	'S'
TLC39	'T'
ULTRACODE	'U'
CODABLOCK_A	'V'
POSICODE	'W'
GRID_MATRIX	'X'
NEC25	'Y'
MESA	'Z'
CODABAR	'a'
CODE39	'b'
UPCA	'c'
EAN13	'd'
I25	'e'
S25 (2BAR and 3BAR)	'f'
MSI	'g'
CODE11	'h'
CODE93	'i'
CODE128	'j'
UNUSED	'k'
CODE49	'l'
M25	'm'
PLESSEY	'n'
CODE16K	'o'
CHANNELCODE	'p'
CODABLOCK_F	'q'
PDF417	'r'
QRCODE	's'
MICROQR_ALT	'-'

Symbology	Code Id
TELEPEN	't'
CODEZ	'u'
VERICODE	'v'
DATAMATRIX	'w'
MAXICODE	'x'
RSS	'y'
GS1_DATABAR	'y'
GS1_DATABAR_LIM	'{'
GS1_DATABAR_EXP	'}'
COMPOSITE	'y'
AZTEC_CODE	'z'

Code Snippet: MainActivity

```
package com.honeywell.sample.intentapisample;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    private static final String TAG = "IntentApiSample";
    private static final String ACTION_BARCODE_DATA = "com.honeywell.sample.action.BARCODE_DATA";

    /**
     * Honeywell DataCollection Intent API
     * Claim scanner
     * Permissions:
     *      "com.honeywell.decode.permission.DECODE"
     */
    private static final String ACTION_CLAIM_SCANNER =
"com.honeywell.aidc.action.ACTION_CLAIM_SCANNER";

    /**
     * Honeywell DataCollection Intent API
     * Release scanner claim
     * Permissions:
     *      "com.honeywell.decode.permission.DECODE"
     */
    private static final String ACTION_RELEASE_SCANNER =
"com.honeywell.aidc.action.ACTION_RELEASE_SCANNER";

    /**
     * Honeywell DataCollection Intent API
     * Optional. Sets the scanner to claim. If scanner is not available or if extra is not used,
     * DataCollection will choose an available scanner.
     * Values : String
     *      "dcs.scanner.imager" : Uses the internal scanner
     *      "dcs.scanner.ring" : Uses the external ring scanner
     */
    private static final String EXTRA_SCANNER = "com.honeywell.aidc.extra.EXTRA_SCANNER";

    /**
     * Honeywell DataCollection Intent API
     * Optional. Sets the profile to use. If profile is not available or if extra is not used,
     * the scanner will use factory default properties (not "DEFAULT" profile properties).
     * Values : String
     */
    private static final String EXTRA_PROFILE = "com.honeywell.aidc.extra.EXTRA_PROFILE";

    /**
     * Honeywell DataCollection Intent API
     * Optional. Overrides the profile properties (non-persistent) until the next scanner claim.
     * Values : Bundle
     */
    private static final String EXTRA_PROPERTIES = "com.honeywell.aidc.extra.EXTRA_PROPERTIES";

    private TextView textView;

    private BroadcastReceiver barcodeDataReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            if (ACTION_BARCODE_DATA.equals(intent.getAction())) {
                /*
                These extras are available:
                    "version" (int) = Data Intent Api version
                    "aimId" (String) = The AIM Identifier
```

```
                "charset" (String) = The charset used to convert "dataBytes" to "data" string
                "codeId" (String) = The Honeywell Symbology Identifier
                "data" (String) = The barcode data as a string
                "dataBytes" (byte[]) = The barcode data as a byte array
                "timestamp" (String) = The barcode timestamp
             */

            int version = intent.getIntExtra("version", 0);
            if (version >= 1) {
                String aimId = intent.getStringExtra("aimId");
                String charset = intent.getStringExtra("charset");
                String codeId = intent.getStringExtra("codeId");
                String data = intent.getStringExtra("data");
                byte[] dataBytes = intent.getByteArrayExtra("dataBytes");
                String dataBytesStr = bytesToHexString(dataBytes);
                String timestamp = intent.getStringExtra("timestamp");

                String text = String.format(
                        "Data:%s\n" +
                        "Charset:%s\n" +
                        "Bytes:%s\n" +
                        "AimId:%s\n" +
                        "CodeId:%s\n" +
                        "Timestamp:%s\n",
                        data, charset, dataBytesStr, aimId, codeId, timestamp);
                setText(text);
            }
        }
    }
};

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    textView = (TextView) findViewById(R.id.textView);
}

@Override
protected void onResume() {
    super.onResume();
    registerReceiver(barcodeDataReceiver, new IntentFilter(ACTION_BARCODE_DATA));
    claimScanner();
}

@Override
protected void onPause() {
    super.onPause();
    unregisterReceiver(barcodeDataReceiver);
    releaseScanner();
}

private void claimScanner() {
    Bundle properties = new Bundle();

    properties.putBoolean("DPR_DATA_INTENT", true);
    properties.putString("DPR_DATA_INTENT_ACTION", ACTION_BARCODE_DATA);

    sendBroadcast(new Intent(ACTION_CLAIM_SCANNER)
                    .putExtra(EXTRA_SCANNER, "dcs.scanner.imager")
                    .putExtra(EXTRA_PROFILE, "MyProfile1")
                    .putExtra(EXTRA_PROPERTIES, properties)
    );
}

private void releaseScanner() {
    sendBroadcast(new Intent(ACTION_RELEASE_SCANNER));
}
```

```
    private void setText(final String text) {
        if (textView != null) {
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    textView.setText(text);
                }
            });
        }
    }

    private String bytesToHexString(byte[] arr) {
        String s = "[]";
        if (arr != null) {
            s = "[";
            for (int i = 0; i < arr.length; i++) {
                s += "0x" + Integer.toHexString(arr[i]) + ", ";
            }
            s = s.substring(0, s.length() - 2) + "]";
        }
        return s;
    }
}
```