## //inside invoke method under index.js

var success = function (message) {

alert(message);

}

var failure = function () {

alert("Error calling Plugin");

}

reader.scan("TEST", success, failure);

## plugin.xml

<?xml version="1.0" encoding="utf-8"?>

<plugin xmlns="http://www.phonegap.com/ns/plugins/1.0"

xmlns:android="http://schemas.android.com/apk/res/android"

id="cordova-plugin-rfidreader"

version="1.0.0">

<name>RFID Reader</name>

<description>Use this plugin to scan the RFID tag in your Cordova/PhoneGap app.</description>

<author>Jiang Jie</author>

<keywords>rfid</keywords>

<license>SIMtech 1.0</license>

<engines>

<engine name="cordova" version=">=3.4.0"/>

</engines>

<asset src="www/reader.js" target="js/reader.js"/>

<js-module src="www/reader.js" name="reader">

<clobbers target="reader" />

</js-module>

<platform name="android">

<config-file target="res/xml/config.xml" parent="/\*">

<feature name="Reader">

<param name="android-package" value="com.reader.plugin.Reader"/>

</feature>

</config-file>

<source-file src="src/android/aretepoplib.jar" target-dir="libs" framework="true" />

<source-file src="src/android/Reader.java" target-dir="src/com/reader/plugin/"/>

<config-file target="AndroidManifest.xml" parent="/\*">

<uses-permission android:name="android.permission.RECORD\_AUDIO" />

<uses-permission android:name="android.permission.MODIFY\_AUDIO\_SETTINGS" />

<uses-permission android:name="android.permission.ACTION\_HEADSET\_PLUG" />

</config-file>

</platform>

</plugin>

## //wwww/reader.js

/\*global cordova, module\*/

module.exports = {

scan: function (name, successCallback, errorCallback) {

cordova.exec(successCallback, errorCallback, "Reader", "scan", [name]);

}

};

## src/android/{aretepoplib.jar, Reader.java}

package com.reader.plugin;

import org.apache.cordova.\*;

import org.json.JSONArray;

import org.json.JSONException;

import com.phychips.rcp.\*;

public class Reader extends CordovaPlugin implements iRcpEvent2 {

String tag\_id="";

@Override

public boolean execute(String action, JSONArray data, CallbackContext callbackContext) throws JSONException {

if (action.equals("scan")) {

RcpApi2.getInstance().setOnRcpEventListener(this);

RcpApi2.getInstance().open();

if (RcpApi2.getInstance().isOpen()) {

RcpApi2.getInstance().startReadTagsWithTid(1, 1, 1);

}

String message = data.getString(0);

callbackContext.success(tag\_id);

return true;

} else {

return false;

}

}

@Override

public void onResetReceived() {

}

@Override

public void onSuccessReceived(int[] ints, int i) {

}

@Override

public void onFailureReceived(int[] ints) {

}

@Override

public void onTagReceived(int[] ints) {

}

@Override

public void onTagWithTidReceived(int[] ints, int[] ints1) {

tag\_id = "";

for (int x = 0; x < ints.length; x++) {

tag\_id = tag\_id + String.valueOf(ints[x]);

}

}

@Override

public void onTagWithRssiReceived(int[] ints, int i) {

}

@Override

public void onReaderInfoReceived(int[] ints) {

}

@Override

public void onRegionReceived(int i) {

}

@Override

public void onSelectParamReceived(int[] ints) {

}

@Override

public void onQueryParamReceived(int[] ints) {

}

@Override

public void onChannelReceived(int i, int i1) {

}

@Override

public void onFhLbtReceived(int[] ints) {

}

@Override

public void onTxPowerLevelReceived(int i) {

}

@Override

public void onTagMemoryReceived(int[] ints) {

}

@Override

public void onTagMemoryLongReceived(int[] ints) {

}

@Override

public void onBatteryStateReceived(int[] ints) {

}

@Override

public void onSessionReceived(int i) {

}

@Override

public void onGenericTransportReceived(int i, int[] ints) {

}

}