AN11690

NXPNCI Android Porting Guidelines

Rev. 1.2 — 19 August 2016
333212

Document information

Info	Content
Keywords	Android, NFC, NXP, NCI, PN7120, PN7150
Abstract	This note describes how to add support for a NXP NCI based NFC Controller to an Android system

Revision history

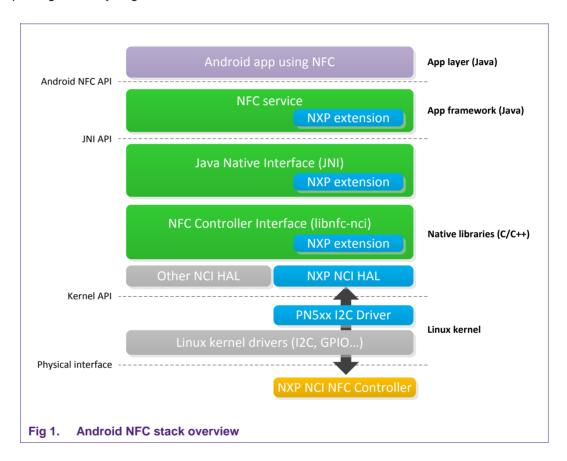| Rev	Date	Description
1.2	20160819	Added support for Android Marshmallow 6.0.1
1.1	20160525	Update for PN7150 support
1.0	20150602	First release

Contact information

For more information, please visit: http://www.nxp.com

AN11690

All information provided in this document is subject to legal disclaimers.

Application note
COMPANY PUBLIC

Rev. 1.2 — 19 August 2016
333212

2 of 23

1. Introduction

This document provides guidelines for the integration of NXP NCI based NFC Controller to an Android platform from software perspective.

It first explains how install the required kernel driver, then it describes step by step how to adapt the Android Open Source Project sources from the NXP-NCI Android NFC package delivery. Fig 1 shows the architecture of the Android NFC stack.

Fig 1.   **Android NFC stack overview**

- The pn5xx_I2c driver is the kernel module allowing to access NXP NCI based NFC Controller hardware resource.

- The NXP NCI HAL module is the implementation of NXP NFC Controller specific Hardware Abstraction Layer.

- The libnfc-nci is the native library providing NFC functionality for which extension is added to support NXP proprietary features (e.g. MIFARE classic support).

- The JNI is a glue code between Java and Native classes. Extension exposes related additional interface.

- The NFC service is the application framework module providing access to NFC functionality. Extension is delivered to support NXP proprietary features.

AN11690

Application note
COMPANY PUBLIC

Rev. 1.2 — 19 August 2016
333212

3 of 23

2. Kernel driver

The NXP-NCI Android stack uses PN5xx I2C kernel mode driver to communicate with the NXP NCI NFC Controller. It is available from the following repository:
www.nxp.com/redirect/github.com/NXPNFCLinux/linux_libnfc-nci

2.1 Driver details

The PN5xx I2C driver offers communication to the NFC Controller connected over I2C physical interface. This is insure through the device node named */dev/pn544*. This driver is compatible with a large range of NXP's NFC Controllers (e.g. PN544).

2.2 Installation instructions

The following instructions assume the driver being installed under the *drivers/misc* kernel source sub-folder. Below instructions may have to be adapted accordingly in case another path is chosen for the driver installation.

2.2.1 Getting the driver

Clone the nxp-pn5xx repository into the kernel directory:

```
$ cd drivers/misc
$ git clone https://github.com/NXPNFCLinux/nxp-pn5xx.git
```

This will create the sub-folder *nxp-pn5xx* containing the following files:

- *pn5xx_i2c.c*: driver implementation
- *pn5xx_i2c.h*: driver interface definition
- *README.md*: repository comments
- *Makefile*: driver related makefile
- *Kconfig*: driver related config file
- *LICENSE*: driver licensing terms
- *sample_devicetree.txt*: example of device tree definition

2.2.2 Including the driver to the kernel

Include the driver to the compilation by adding below line to the heading makefile (*drivers/misc/Makefile*).

```
obj-y          += nxp-pn5xx/
```

Include the driver config by adding below line to the heading configuration file (*drivers/misc/Kconfig*).

```
source "drivers/misc/nxp-pn5xx/Kconfig"
```

2.2.3 Creating the device node

Two methods are supported for the creation of the */dev/pn544* device node: device tree and platform data. Any of the two methods can be used, but of course the I2C address (0x28 in the below examples) and GPIO assignments must be adapted to the hardware integration in the platform.

2.2.3.1 Device tree

Below is an example of definition to be added to the platform device tree file (.*dts* file located for instance under *arch/arm/boot/dts* kernel sub-folder for ARM based platform).

```
&i2c{
    status = "okay";
    pn547: pn547@28 {
        compatible = "nxp,pn547";
        reg = <0x28>;
        clock-frequency = <400000>;
        interrupt-gpios = <&gpio2 17 0>;
        enable-gpios = <&gpio4 21 0>;
    };
};
```

2.2.3.2 Platform data

Below is an example of definition to be added to the platform definition file. The structure *pn544_i2c_platform_data* being defined in the driver interface header file, *pn5xx_i2c.h* must be included in the platform definition file, and *pn5xx_i2c.h* file must be copied to *include/linux* kernel source sub-folder.

```
static struct pn544_i2c_platform_data nfc_pdata = {
    .irq_gpio = GPIO_TO_PIN(1,29),
    .ven_gpio = GPIO_TO_PIN(0,30),
    .firm_gpio = GPIO_UNUSED
    .clkreq_gpio = GPIO_UNUSED
};

static struct i2c_board_info __initdata nfc_board_info[] = {
    {
        I2C_BOARD_INFO("pn547", 0x28),
        .platform_data = &nfc_pdata,
    },
};
```

Then the declared **nfc_board_info** structure must be added to the platform using dedicated procedure (platform specific).

2.2.4 Building the driver

Through *menuconfig* procedure include the driver to the build, as built-in (<*>) or modularizes features (<M>):

```
Device Drivers --->
    Misc devices --->
        < > NXP PN5XX based driver
```

If <M> option is selected, build the driver and install the generated *pn5xx_i2c.ko* module. Otherwise if built-in, rebuild the complete kernel, the driver will be included in the kernel image.

If the device tree method was used in previous step, build the platform related device tree and install generated .dtb file.

3. AOSP adaptation

3.1 Android Marshmallow 6.0.1

Below step-by-step procedure is based on NXP's Android NFC delivery on the following repositories: https://github.com/NXPNFCLinux/nxpnfc_android_marshmallow (later referenced as [NxpNfc_Android_marshmallow]) and https://github.com/NXPNFCProject (later referenced as [NxpNfc_Project]).

3.1.1 Step 1: retrieving NXP-NCI NFC delivery

Retrieve the NXP-NCI NFC Android manifest file from [NxpNfc_Android_marshmallow] using *wget* command:

```
$ wget
https://raw.githubusercontent.com/NXPNFCLinux/nxpnfc_android_marshmallow
/master/nxpnfc_manifest.xml
```

Or using *curl* command:

```
$ curl
https://raw.githubusercontent.com/NXPNFCLinux/nxpnfc_android_marshmallow
/master/nxpnfc_manifest.xml > nxpnfc_manifest.xml
```

Then install it as local manifest in the AOSP source directory:

```
& mv nxpnfc_manifest.xml {ANDROID_BUILD_TOP}/.repo/local_manifests/
```

And apply changes brought by NXP-NCI NFC Android manifest:

```
$ repo sync --force-sync
```

This will autonomously:

- Replace original AOSP *external/libnfc-nci* folder with the one from [NxpNfc_Project]/NFC_NCIHAL_libnfc-nci
- Replace original AOSP *packages/apps/Nfc* folder with the one from [NxpNfc_Project]/NFC_NCIHAL_Nfc and [NxpNfc_Project]/NXPNFC_Reference
- Retrieve source code to be merge with AOSP in next step 2 from [NxpNfc_Project]/NFC_NCIHAL_base
- Retrieve installation scripts, patches to be used in next step 2 and configuration files from [NxpNfc_Android_marshmallow]

3.1.2 Step 2: installing NXP-NCI delivery

Run the installation script (with either PN7120 or PN7150 as <NFCC> parameter):

```
$ ./NxpNfcAndroid/install.sh <NFCC>
```

This will autonomously:

- Replace AOSP *hardware/libhardware/include/hardware/nfc.h* file with the one from NxpNfcAndroid sub-folder created at step 1

- Replace AOSP *frameworks/base/core/java/android/nfc* folder with the one from NxpNfcAndroid sub-folder created at step 1

- Copy AOSP *frameworks/base/core/java/com/nxp* folder from NxpNfcAndroid sub-folder created at step 1

- Copy AOSP *frameworks/base/core/java/com/vzw* folder from NxpNfcAndroid sub-folder created at step 1

- Patch the AOSP *frameworks/base/Android.mk* to insert NXP additions

- Patch the AOSP *frameworks/base/api/current.txt, frameworks/base/api/system-current.txt, frameworks/base/core/java/android/content/pm/PackageManager.java* and *frameworks/base/core/res/res/values/attrs.xml* to add FeliCa HCE support in the scope of PN7150

- Patch *external/libnfc-nci/halimpl/pn54x/Android.mk* and *external/libnfc-nci/Android.mk* to set support of PN7150 or PN7120

- Patch *external/libnfc-nci/halimpl/pn54x/hal/phNxpNciHal.c* in specific case of PN7120

- Patch *packages/apps/Nfc/nci/jni/Android.mk* to set support of PN7150 or PN7120

3.1.3 Step 3: updating configuration files

Adapt the *libnfc-brcm.conf* and *libnfc-nxp.conf* files located in *NxpNfcAndroid/conf* sub-folder, created at step 1, according to the integration specificities.

For instance if using a system clock instead of an on-board crystal, the value of parameter "NXP_SYS_CLK_SRC_SEL" in *libnfc-nxp.conf* must reflect this configuration.

More details about the configuration files can be find in chapter "4 Configuration Files".

3.1.4 Step 4: adding NFC to the build

In the *device.mk* makefile (e.g. *device/**brand/platform**/device.mk*)

- Add the NFC related packages to the android build

```
# NFC packages
PRODUCT_PACKAGES += \
          libnfc-nci \
          libnfc_nci_jni \
          nfc_nci_pn54x.default\
          NfcNci \
          Tag \
          com.android.nfc_extras
```

- Add xml files to Android launches the NFC functionalities:

```
PRODUCT_COPY_FILES += \
frameworks/native/data/etc/com.nxp.mifare.xml:system/etc/permissions/com
.nxp.mifare.xml \
frameworks/native/data/etc/com.android.nfc_extras.xml:system/etc/permiss
ions/com.android.nfc_extras.xml \
frameworks/native/data/etc/android.hardware.nfc.xml:system/etc/permissio
ns/android.hardware.nfc.xml \
frameworks/native/data/etc/android.hardware.nfc.hce.xml:system/etc/permi
ssions/android.hardware.nfc.hce.xml \
NxpNfcAndroid/android.hardware.nfc.hcef.xml:system/etc/permissions/andro
id.hardware.nfc.hcef.xml \
NxpNfcAndroid/conf/libnfc-brcm.conf:system/etc/libnfc-brcm.conf \
NxpNfcAndroid/conf/libnfc-nxp.conf:system/etc/libnfc-nxp.conf
```

3.1.5 Step 5: changing device owner and permissions

On the *system/core/rootdir/init.rc* file, add the following lines to the end of the *on boot* section:

```
# NFC
    setprop ro.nfc.port "I2C"
    chmod 0660 /dev/pn544
    chown nfc nfc /dev/pn544
```

3.1.6 Step 6: building and installing NFC

Build and flash the system image (the boot image shall already contains the kernel driver as instructed in chapter "2. Kernel driver").

3.1.7 Step 7: verifying NFC functionality

In "Settings" app check NFC is ON. NFC functionality should be then up and running, ready to discover NFC tags or exchange data with remote NFC devices.

AN11690

Application note
COMPANY PUBLIC

Rev. 1.2 — 19 August 2016
333212

9 of 23

3.2 Android Lollipop 5.1.1

Below step-by-step procedure is based on NXP's Android NFC delivery on the following repositories: https://github.com/NXPNFCLinux/nxpnfc_android_lollipop (later referenced as [NxpNfc_Android_lollipop]) and https://github.com/NXPNFCProject (later referenced as [NxpNfc_Project]).

3.2.1 Step 1: retrieving NXP-NCI NFC delivery

Retrieve the NXP-NCI NFC Android manifest file from [NxpNfc_Android_lollipop] using *wget* command:

```
$ wget
https://raw.githubusercontent.com/NXPNFCLinux/nxpnfc_android_lollipop/ma
ster/nxpnfc_manifest.xml
```

Or using *curl* command:

```
$ curl
https://raw.githubusercontent.com/NXPNFCLinux/nxpnfc_android_lollipop/ma
ster/nxpnfc_manifest.xml > nxpnfc_manifest.xml
```

Then install it as local manifest in the AOSP source directory:

```
& mv nxpnfc_manifest.xml {ANDROID_BUILD_TOP}/.repo/local_manifests/
```

And apply changes brought by NXP-NCI NFC Android manifest:

```
$ repo sync --force-sync
```

This will autonomously:

- Replace original AOSP *external/libnfc-nci* folder with the one from [NxpNfc_Project]/NFC_NCIHAL_libnfc-nci
- Replace original AOSP *packages/apps/Nfc* folder with the one from [NxpNfc_Project]/NFC_NCIHAL_Nfc and [NxpNfc_Project]/NXPNFC_Reference
- Retrieve source code to be merge with AOSP in next step 2 from [NxpNfc_Project]/NFC_NCIHAL_base
- Retrieve installation scripts, patches to be used in next step 2 and configuration files from [NxpNfc_Android_lollipop]

3.2.2 Step 2: installing NXP-NCI delivery

Run the installation script (with either PN7120 or PN7150 as <NFCC> parameter):

```
$ ./NxpNfcAndroid/install.sh <NFCC>
```

This will autonomously:

- Replace AOSP *hardware/libhardware/include/hardware/nfc.h* file with the one from NxpNfcAndroid sub-folder created at step 1
- Replace AOSP *frameworks/base/core/java/android/nfc* folder with the one from NxpNfcAndroid sub-folder created at step 1
- Copy AOSP *frameworks/base/core/java/com/nxp* folder from NxpNfcAndroid sub-folder created at step 1
- Copy AOSP *frameworks/base/core/java/com/vzw* folder from NxpNfcAndroid sub-folder created at step 1
- Patch the AOSP *frameworks/base/Android.mk* to insert NXP additions
- Patch *external/libnfc-nci/halimpl/pn54x/Android.mk* and *external/libnfc-nci/Android.mk* to set support of PN7120 if selected as script parameter
- Patch *packages/apps/Nfc/nci/jni/Android.mk* to set support of PN7120 if selected as script parameter

3.2.3   Step 3: updating configuration files

Adapt the *libnfc-brcm.conf* and *libnfc-nxp.conf* files located in *NxpNfcAndroid/conf* sub-folder, created at step 1, according to the integration specificities.

For instance if using a system clock instead of an on-board crystal, the value of parameter "NXP_SYS_CLK_SRC_SEL" in *libnfc-nxp.conf* must reflect this configuration.

More details about the configuration files can be find in chapter "4 Configuration Files".

3.2.4   Step 4: adding NFC to the build

In the *device.mk* makefile (e.g. *device/**brand**/**platform**/device.mk*)

- Add the NFC related packages to the android build

```
# NFC packages
PRODUCT_PACKAGES += \
        libnfc-nci \
        libnfc_nci_jni \
        nfc_nci_pn54x.default\
        NfcNci \
        Tag \
        com.android.nfc_extras
```

- Add xml files to Android launches the NFC functionalities:

```
PRODUCT_COPY_FILES += \
frameworks/native/data/etc/com.nxp.mifare.xml:system/etc/permissions/com
.nxp.mifare.xml \
frameworks/native/data/etc/com.android.nfc_extras.xml:system/etc/permiss
ions/com.android.nfc_extras.xml \
```

```
frameworks/native/data/etc/android.hardware.nfc.xml:system/etc/permissio
ns/android.hardware.nfc.xml \
frameworks/native/data/etc/android.hardware.nfc.hce.xml:system/etc/permi
ssions/android.hardware.nfc.hce.xml \
NxpNfcAndroid/conf/libnfc-brcm.conf:system/etc/libnfc-brcm.conf \
NxpNfcAndroid/conf/libnfc-nxp.conf:system/etc/libnfc-nxp.conf
```

3.2.5 Step 5: changing device owner and permissions

On the *system/core/rootdir/init.rc* file, add the following lines to the end of the *on boot* section:

```
# NFC
    setprop ro.nfc.port "I2C"
    chmod 0660 /dev/pn544
    chown nfc nfc /dev/pn544
```

3.2.6 Step 6: building and installing NFC

Build and flash the system image (the boot image shall already contains the kernel driver as instructed in chapter "2. Kernel driver").

3.2.7 Step 7: verifying NFC functionality

In "Settings" app check NFC is ON. NFC functionality should be then up and running, ready to discover NFC tags or exchange data with remote NFC devices.

AN11690

Application note
COMPANY PUBLIC

Rev. 1.2 — 19 August 2016
333212

12 of 23

3.3 Android KitKat 4.4.4

Below step-by-step procedure is based on NXP-NCI Android NFC package delivered by NXP on the following repository: https://github.com/NXPNFCLinux/nxpnfc_android_kitkat.

3.3.1 Step 1: getting the release package

Clone the related repository:

```
$ git clone https://github.com/NXPNFCLinux/nxpnfc_android_kk.git
```

The following directory structure will be created:

```
├── aosp
│   └── external/libnfc-nci
│       └── ...
│   └── frameworks/base
│       └── core/java/android/nfc
│           └── ...
│       └── core/java/com/vzw/nfc
│           └── ...
│       └── Android.mk
│   └── hardware/libhardware/include/hardware
│       └── nfc.h
│   └── packages/apps/Nfc
│       └── ...
├── conf
│   └── libnfc-brcm.conf
│   └── libnfc-nxp.conf
├── doc
│   └── AN11690 – NXP-NCI Android Porting Guidelines.pdf
└── README.txt
```

3.3.2 Step 2: merging files

Merge the files from the NXP-NCI Android NFC package (*aosp* sub-folder) into the target AOSP source directory:

- Replace original AOSP *external/libnfc-nci* folder
- Replace original AOSP *frameworks/base/core/java/android/nfc* folder
- Copy or replace original AOSP *frameworks/base/core/java/com/vzw/nfc* folder
- Merge original AOSP *frameworks/base/Android.mk* with the one from the delivery (only "nfc" related items must be added to the original makefile)
- Replace original AOSP *hardware/libhardware/include/hardware/nfc.h* file
- Replace original AOSP *packages/apps/Nfc* folder

3.3.3   Step 3: selecting the NFC Controller

In the following makefiles:

- *aosp/external/libnfc-nci/Android.mk*

- *aosp/external/libnfc-nci/halimpl/pn54x/Android.mk*

- *aosp/packages/apps/Nfc/nci/jni/Android.mk*

Adapt the following line according to the integrated NFC Controller:

➢ for PN7150

```
D_CFLAGS += -DNFC_NXP_CHIP_PN548AD=TRUE
```

➢ for PN7120

```
D_CFLAGS += -DNFC_NXP_CHIP_PN548AD=FALSE
```

3.3.4   Step 4: adding NFC to the build

In the *device.mk* makefile (e.g. *device/**brand**/**platform**/device.mk*)

- Add the NFC related packages to the android build

```
# NFC packages
PRODUCT_PACKAGES += \
        libnfc-nci \
        libnfc_nci_jni \
        nfc_nci_pn54x.default\
        NfcNci \
        Tag \
        com.android.nfc_extras
```

- Add xml files to Android launches the NFC functionalities:

```
PRODUCT_COPY_FILES += \
frameworks/native/data/etc/com.nxp.mifare.xml:system/etc/permissions/com
.nxp.mifare.xml \
frameworks/native/data/etc/com.android.nfc_extras.xml:system/etc/permiss
ions/com.android.nfc_extras.xml \
frameworks/native/data/etc/android.hardware.nfc.xml:system/etc/permissio
ns/android.hardware.nfc.xml \
frameworks/native/data/etc/android.hardware.nfc.hce.xml:system/etc/permi
ssions/android.hardware.nfc.hce.xml
```

3.3.5   Step 5: changing device owner and permissions

On the *system/core/rootdir/init.rc* file, add the following lines to the end of the *on boot* section:

```
# NFC
setprop ro.nfc.port "I2C"
chmod 0660 /dev/pn544
chown nfc nfc /dev/pn544
```

3.3.6 Step 6: building and installing NFC

Build and flash the system image (the boot image shall already contains the kernel driver as instructed in chapter "2. Kernel driver").

Once the Android platform boots up, add the 2 configuration files required by the libnfc-nci library:

```
$ adb push libnfc-brcm.conf /etc/
$ adb push libnfc-nxp.conf /etc/
```

Examples are given in the NXP-NCI Android NFC package, under *conf* sub-directory, but pay attention that some adaptation may be required according to your integration (see chapter "4 Configuration Files" for more details).Then reboot the platform.

3.3.7 Step 7: verifying NFC functionality

In "Settings" app check NFC is ON.

NFC functionality should be then up and running, ready to discover NFC tags or exchange data with remote NFC devices.

3.4 Others Android versions

For other Android versions others than KitKat 4.4.4, Lollipop 5.1.1 or Marshmallow 6.0.1, the AOSP must be manually adapted (merged) from the source retrieved as detailed in chapters 3.3, 3.2 or 3.1 depending of the targeted version.

4. Configuration Files

Two files allow configuring the libnfc-nci library at runtime: *libnfc-brcm.conf* and *libnfc-nxp.conf*. There are defining tags which are impacting library behavior. The value of the tags depends on the NFC Controller IC and the targeted platform. For more details, refer to the examples given in *conf* sub-folder of the stack delivery (see chapter 3.1.3 or 3.3.1).

These files are loaded by the library, from */etc* directory of the target, during the initialization phase.

Pay attention that the configuration files provided as example relates to the NFC Controller demo boards. These files must be adapted according to the targeted integration.

Below is the description of the different useful tags in the configuration files (refer to the conf files for detailed information about the tag values).

Table 1. Tag list of libnfc-brcm.conf file

Tag	Description
APPL_TRACE_LEVEL	Log levels for libnfc-nci. Recommended value for debugging is 0xFF.
PROTOCOL_TRACE_LEVEL	Log levels for libnfc-nci. Recommended value for debugging is 0xFF.
HOST_LISTEN_ENABLE	Configuration force HOST listen feature.
SCREEN_OFF_POWER_STATE	Configuration of screen off power state.
POLLING_TECH_MASK	Configuration of the polling technologies.
P2P_LISTEN_TECH_MASK	Configuration of listen technologies for P2P.
NFA_DM_DISC_DURATION_POLL	Configuration of the discovery loop TOTAL DURATION (in milliseconds)
NFA_MAX_EE_SUPPORTED	Set the maximum number of Execution Environments supported.

Table 2. Tag list of libnfc-nxp.conf file

Tag	Description
NXPLOG_EXTNS_LOGLEVEL	Configure level of EXTNS logs. Recommended value for debug is 0x03.
NXPLOG_NCIHAL_LOGLEVEL	Set level of NCIHAL logs. Recommended value for debug is 0x03.
NXPLOG_NCIX_LOGLEVEL	Set level of NCIX logs. Recommended value for debug is 0x03.
NXPLOG_NCIR_LOGLEVEL	Set level of NCIR logs. Recommended value for debug is 0x03.
NXPLOG_FWDNLD_LOGLEVEL	Set level of FWDNLD logs. Recommended value for debug is 0x03.

Tag	Description
NXPLOG_TML_LOGLEVEL	Set level of FWDNLD logs. Recommended value for debug is 0x03.
MIFARE_READER_ENABLE	Set the support of MIFARE Classic Reader.
NXP_SYS_CLK_SRC_SEL	Configure the clock source of the NFC Controller.
NXP_SYS_CLK_FREQ_SEL	Set the clock frequency in case of PLL clock source.
NXP_SYS_CLOCK_TO_CFG	Set clock request acknowledgment time value in case of PLL clock source.
NXP_ACT_PROP_EXTN	Set NXP's NFC Controller proprietary features.
NXP_NFC_MERGE_RF_PARAMS	Set NFCC Configuration control
NXP_NFC_PROFILE_EXTN	Set discovery profile.
NXP_RF_CONF_BLK_x	Set platform specific RF configuration.
NXP_CORE_CONF_EXTN	Configure proprietary parts of the NFC Controller.
NXP_CORE_CONF	Configure standardized parts of the NFC Controller.
NXP_CORE_MFCKEY_SETTING	Proprietary configuration for MIFARE Key storage.

5. Troubleshooting

The following items may help figuring out what is going wrong in case NFC is not working as expected when starting the Android device.

5.1 Device node rights

The following ADB logs may indicate a wrong setting of the kernel module related device node access rights:

```
…
D/NxpTml  (  894): Opening port=/dev/pn544
E/NxpTml  (  894): _i2c_open() Failed: retval ffffffff
D/NxpTml  (  894): phTmlNfc_i2c_reset(), VEN level 0
E/NxpHal  (  894): phTmlNfc_Init Failed
D/NfcAdaptation(  894): NfcAdaptation::HalDeviceContextCallback: event=0
I/BrcmNfcNfa(  894): nfc_main_hal_cback event: HAL_NFC_OPEN_CPLT_EVT(0x0), status=1
D/NxpHal  (  894): Failed to deallocate (list empty)
D/NxpHal  (  894): Node dump:
D/NxpHal  (  894): Failed to deallocate (list empty)
D/NxpHal  (  894): Node dump:
…
```

The pn5xx_i2c device node should usually appear with the following rights:

```
$ adb shell ls -als /dev/pn544
crw-rw---- nfc       nfc      10,  54 2016-05-03 13:05 pn544
```

If this is not the case, refer to related procedure chapter 3.1.5, 3.2.5 or 3.3.5.

5.2 Configuration files

The following ADB logs may indicates the absence of the configuration files:

```
…
D/BrcmNfcJni(  893): PowerSwitch::initialize: level=PS-FULL (1)
D/NfcAdaptation(  893): bool CNfcConfig::readConfig(const char*, bool) Cannot open config file
/etc/libnfc-brcm.conf
D/NfcAdaptation(  893): bool CNfcConfig::readConfig(const char*, bool) Using default value for all
settings
D/BrcmNfcJni(  893): PowerSwitch::initialize: desired screen-off state=1
D/NfcAdaptation(  893): NfcAdaptation::Initialize: enter
E/NfcAdaptation(  893): NfcAdaptation::Initialize: ver=NFCDROID-AOSP_L_00.01 nfa=NFA_PI_1.03.66+
D/BrcmNfcJni(  893): initializeGlobalAppLogLevel: level=5
D/NfcAdaptation(  893): NfcAdaptation::NFCA_TASK: enter
I/BrcmNfcNfa(  893): GKI_run(): Start/Stop GKI_timer_update_registered!
D/NfcAdaptation(  893): NfcAdaptation::Thread: enter
I/BrcmNfcNfa(  893): NFC_TASK started.
```

```
D/NfcAdaptation(  893): NfcAdaptation::InitializeHalDeviceContext: enter
E/NfcAdaptation(  893): No HAL module specified in config, falling back to BCM2079x
E/NfcAdaptation(  893): NfcAdaptation::InitializeHalDeviceContext: fail hw_get_module
nfc_nci.bcm2079x
D/NfcAdaptation(  893): NfcAdaptation::InitializeHalDeviceContext: exit
D/NfcAdaptation(  893): NfcAdaptation::Initialize: exit
I/BrcmNfcNfa(  893): NFA_Init ()
I/BrcmNfcNfa(  893): nfa_dm_init ()
…
```

They should usually be installed in */etc* android target directory:

```
$ adb shell ls -als /etc/libnfc*.conf
-rw-r--r-- root      root          3146 2016-03-11 23:07 libnfc-brcm.conf
-rw-r--r-- root      root          3738 2016-03-11 23:07 libnfc-nxp.conf
```

If this is not the case, refer to related procedure 3.1.4, 3.2.4 or 3.3.6.

5.3 NXP's NFC library

The following ADB logs may indicates missing NXP's NFC specific library:

```
…
I/BrcmNfcNfa( 2609): NFC_TASK started.
D/NfcAdaptation( 2609): NfcAdaptation::Thread: exit
D/NfcAdaptation( 2609): NfcAdaptation::InitializeHalDeviceContext: enter
D/NfcAdaptation( 2609): const CNfcParam* CNfcConfig::find(const char*) const found
NCI_HAL_MODULE=nfc_nci.pn54x
E/NfcAdaptation( 2609): NfcAdaptation::InitializeHalDeviceContext: fail hw_get_module nfc_nci.pn54x
D/NfcAdaptation( 2609): NfcAdaptation::InitializeHalDeviceContext: exit
D/NfcAdaptation( 2609): NfcAdaptation::Initialize: exit
…
```

The library should be located under */system/lib/hw* android target sub-directory:

```
$ adb shell ls -als /system/lib/hw/nfc*
-rw-r--r-- root  root   119188 2016-03-14 13:55 nfc_nci.pn54x.default.so
```

If this is not the case, insure it is properly built:

```
$ mmm external/libnfc-nci/ - snod
```

You can then either flash the newly created *system.img* or just copy the library to the android target:

```
$ adb push nfc_nci.pn54x.default.so /system/lib/hw/
```

AN11690

Application note
COMPANY PUBLIC   **Rev. 1.2 — 19 August 2016**
333212   **20 of 23**

5.4 NFC Controller choice

The following ADB logs may indicates a wrong adaptation of the NFC libraries to the NFC controller integrated (PN7120 or PN7150):

```
…
D/NxpTml  (25279): PN54X - Write requested.....
D/NxpTml  (25279): PN54X - Invoking I2C Write.....
D/NxpTml  (25279): PN54X - Read requested.....
D/NxpTml  (25279): PN54X - Invoking I2C Read.....
D/NxpNciX (25279): len =  10 > 20020702310100380101
D/NxpTml  (25279): PN54X - I2C Write successful.....
D/NxpTml  (25279): PN54X - Posting Fresh Write message.....
D/NxpTml  (25279): PN54X - Tml Writer Thread Running...............
D/NxpHal  (25279): write successful status = 0x0
D/NxpTml  (25279): PN54X - I2C Read successful.....
D/NxpNciR (25279): len =   4 > 40020106
D/NxpTml  (25279): PN54X - Posting read message.....
D/NxpHal  (25279): read successful status = 0x0
D/NxpHal  (25279): > Deinit for LLCP set_config 0x0 0x0 0x0
D/NxpHal  (25279): phNxpNciHal_print_res_status: response status =STATUS_OK
…
```

If this is the case, refer to related procedure chapter 3.1.2, 3.2.2 or 3.3.3.

6. Legal information

6.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However NXP Semiconductors does not give any representations or warranties expressed or implied as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect incidental punitive special or consequential damages (including - without limitation - lost profits lost savings business interruption costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence) warranty breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document including without limitation specifications and product descriptions at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed authorized or warranted to be suitable for use in life support life-critical or safety-critical systems or equipment nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default damage costs or problem which is based on any weakness or default in the customer's applications or products or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors its affiliates and their suppliers expressly disclaim all warranties whether express implied or statutory including but not limited to the implied warranties of non-infringement merchantability and fitness for a particular purpose. The entire risk as to the quality or arising out of the use or performance of this product remains with customer.

In no event shall NXP Semiconductors its affiliates or their suppliers be liable to customer for any special indirect consequential punitive or incidental damages (including without limitation damages for loss of business business interruption loss of use loss of data or information and the like) arising out the use of or inability to use the product whether or not based on tort (including negligence) strict liability breach of contract breach of warranty or any other theory even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation all damages referenced above and all direct or general damages) the entire liability of NXP Semiconductors its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations exclusions and disclaimers shall apply to the maximum extent permitted by applicable law even if any remedy fails of its essential purpose.

6.3 Licenses

Purchase of NXP ICs with NFC technology

Purchase of an NXP Semiconductors IC that complies with one of the Near Field Communication (NFC) standards ISO/IEC 18092 and ISO/IEC 21481 does not convey an implied license under any patent right infringed by implementation of any of those standards. Purchase of NXP Semiconductors IC does not include a license to any NXP patent (or other IP right) covering combinations of those products with other products, whether hardware or software.

6.4 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

MIFARE> — is a trademark of NXP Semiconductors N.V.

7. Contents