SN1XX- NFC Host SW Integration Guideline

Rev. 0.3 — 2/26/2021 Application Note

Document information

Info	Content
Keywords	NFC, Android

Revision history

Rev	Date	Description
0.1	2021-02-23	Initial version for Android 12 NXP NFC Host SW Integration Guide
0.2	2021-02-25	Host SW configuration parameters are updated
0.3	2021-02-26	Driver integration steps are updated

Contact information

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

1. Introduction

NXP's NFC controller SN110T/U is designed to work with Android open source. Fig. 1, shows the NXP's development and validation platform setup with Hi-key board 960 & Iguana Lite board.

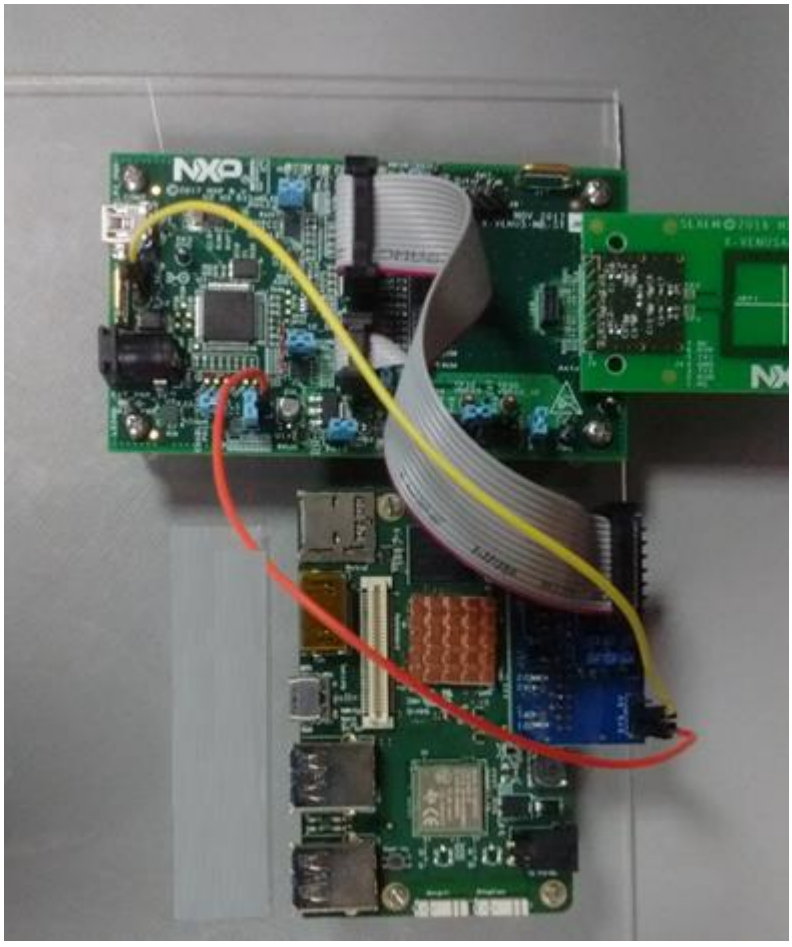


Figure 1: Hikey960 with Iguana Lite Board

2. Abbreviations

NFC	Near Field Communication
OEM	Original Equipment Manufacturer
HW	Hardware
IC	Integrated Circuit
SWP	Single Wire Protocol
GPIO	General Purpose Input / Output
I2C	Inter-Integrated Circuit
SPI	Serial Peripheral Interface
SW	Software
SE	Secure Element
OMAPI	Open Mobile Application Programming Interface
AOSP	Android Open Source Project
HAL	Hardware Abstraction Layer
eSE	Embedded Secure Element
OS	Operating System
SEMS	Secure Element Management Service
LS	Loader Service
GSMA	GSM Association
GSM	Global System for Mobile
NFCC	NFC Controller
SMB	System Mail Box
HIDL	HAL interface definition language
UICC	Universal Integrated Circuit Card
ISO	International Organization for Standardization
P2P	Peer To Peer
DH	Device Host
DTA	Device Test Application
NA	Not Applicable
MPOS	Mobile Point of Sale
TEE	Trusted Execution Environment

3. Scope

This document provides guidelines for setting up NXP's new generation NFC/SE monolithic platform SN110T/U in Android 12 build environment. It is a reference guideline for basic system integration. OEM integration may have variations based on actual system integration.

4. General steps for Android NFC integration

For the NFC software integration with Android, it is hereby assumed that NFC IC HW integration is done in a platform with following checks.

- Schematic reviewed with NXP
- HW IC interface like I2C/SPI, SWP (if used) working.
- Antenna designed and reviewed
- Antenna connection working
- GPIO connections checked

Fig. 2, shows the basic flow for Android NFC SW bring up. Following sections describe these steps in detail.

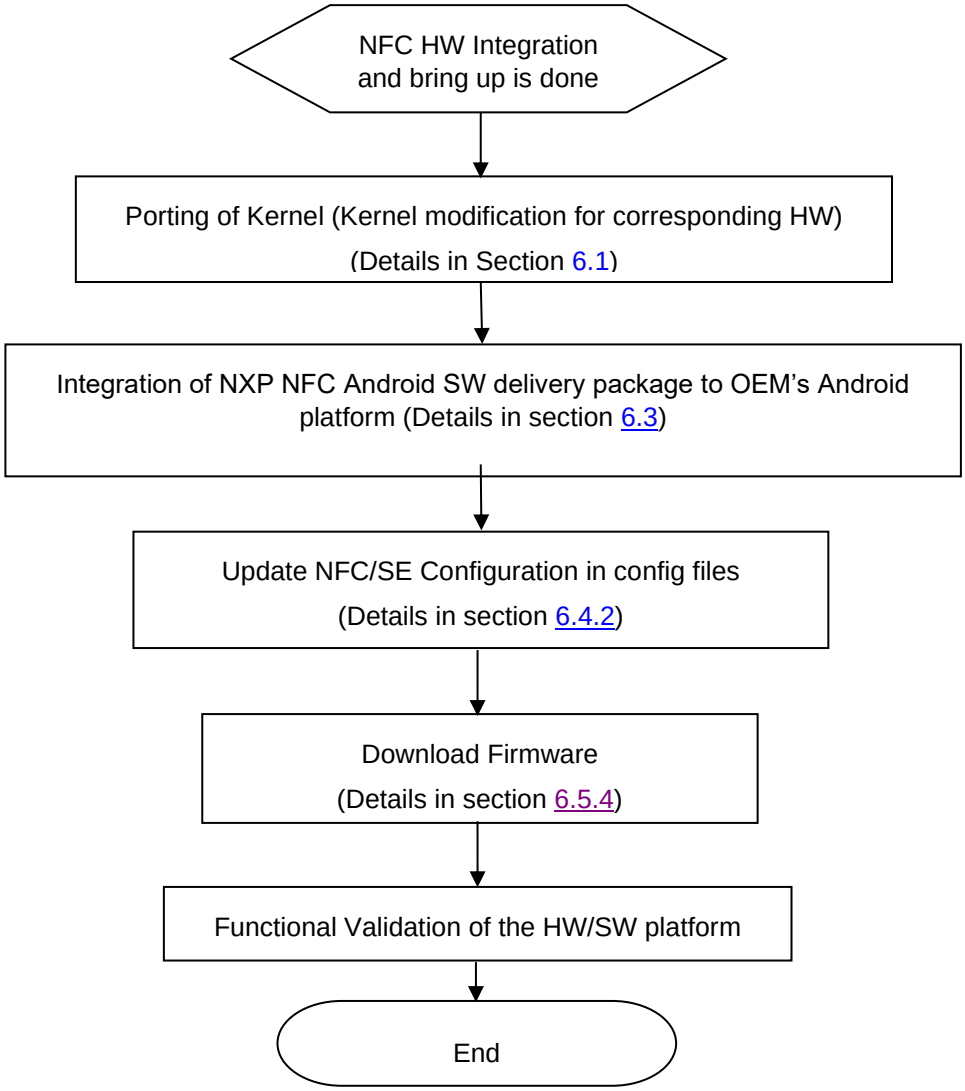


Figure 2: Android NFC SW bring up flow

5. Architecture Overview

Fig. 3, describes the architecture of Android 12 based NXP delivery package.

OMAPI implementation is part of the AOSP from Android P version onwards and NXP does not make any modification in Android OMAPI service layer.

Additionally, NXP implements TEE HAL interface.

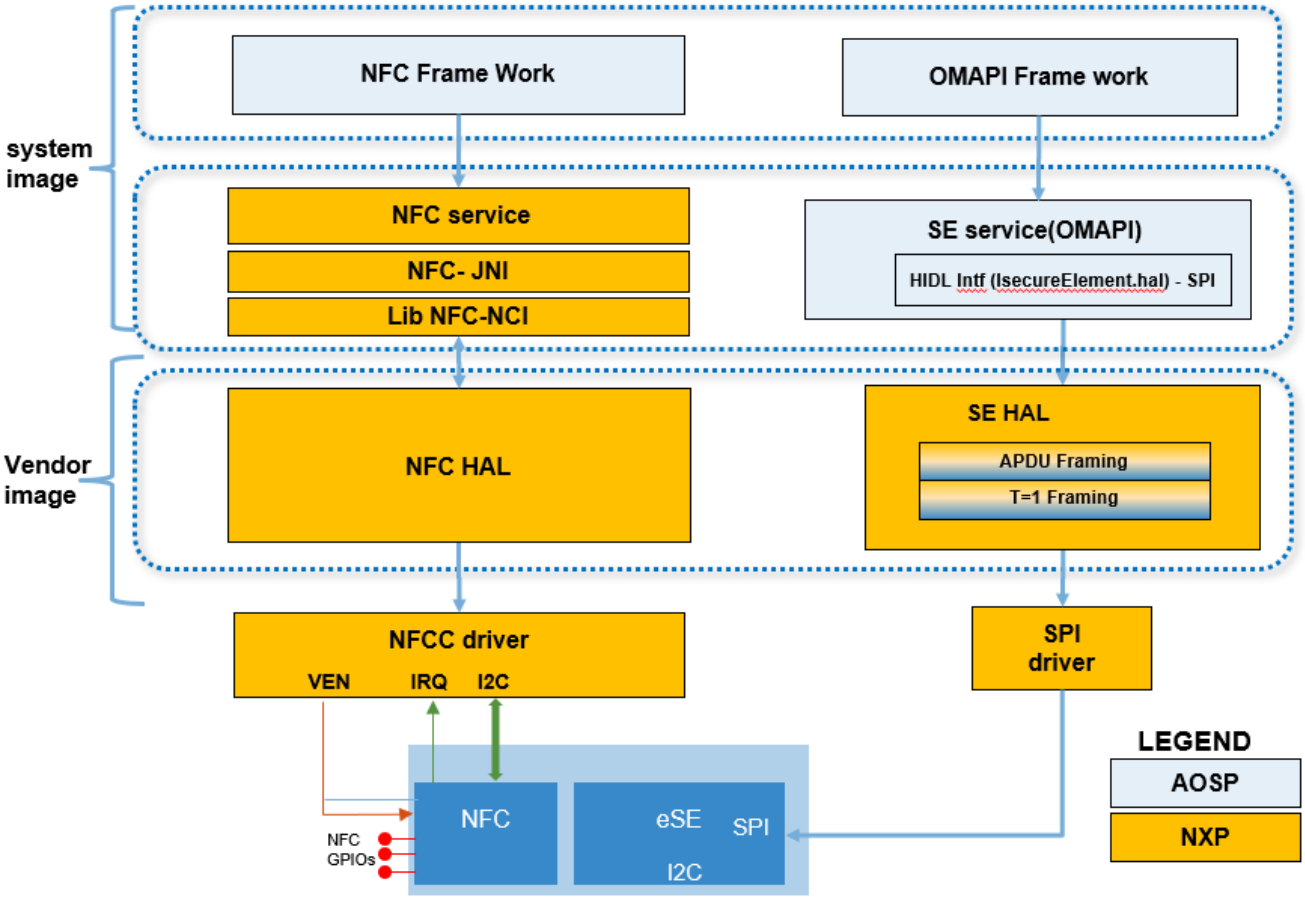


Figure 3: NFC/OMAPI architecture in Android 12

6. Setup of Android NFC

6.1 Kernel driver setup for NXP-NFCC

6.1.1 Android Kernel Preparation

The hikey platform kernel can be downloaded by the below command:

git clone https://android.googlesource.com/kernel/hikey-linaro

Additional information regarding hikey kernel:
git branch: android-hikey-linaro-4.19
git commit: 3c839c5a99d2e92c8978b0965736203d5b545262

Steps to perform in platform's kernel root directory to integrate NXP specific I2C and SPI drivers for accessing NFCC and eSE.

1. Download NFC I2C & SPI drivers from below git hub location:
   https://github.com/NXPNFCProject/NXPNFC_I2CDriver
   https://github.com/NXPNFCProject/NXPESE_SPIDriver
2. Create nxp/pn8xT folder inside kernel/driver/
3. Copy pn54x-i2c from NXPNFC_I2CDriver and keep inside as per chip type
4. Copy p73-spi from NXPNFC_SPIDriver to nxp/pn8xT
5. Add the DTS changes required in your platform DTS file

```
clock-frequency = <1000000>;
        pn544: pn544@28 {
            compatible = "nxp,pn544";
            reg = <0x28>;
            //As per Hi960 mapping ex:GPIO208
            nxp,pn544-irq = <&gpio26 0 0>;
            nxp,pn544-ven = <&gpio26 1 0>;
            nxp,pn544-fw-dwnld = <&gpio26 2 0>;
            nxp,pn544-iso-pwr-rst = <&gpio6 4 0>;
        };
        p61@0 {
            compatible = "nxp,p61";
            reg = <0>;
            nxp,p61-irq = <&gpio2 3 0>;
            nxp,p61-rst = <&gpio2 5 0>;
            nxp,trusted-se = <&gpio26 4 0>;
            spi-max-frequency = <20000000>;
            nxp,nfcc = "2-0028";
        };
```

6. Define the Kernel configurations for the driver in kconfig file:-

```
config NXP_NFC_I2C
tristate "NXP NCI based NFC I2C Slave Driver for SNxxx"
depends on I2C
help
  This enables the NFC driver for SNxxx based devices.
  This is for I2C connected version. NCI protocol logic
  resides in the usermode and it has no other NFC dependencies.

  If unsure, say N.

config NXP_ESE_P73
tristate "Nxp P73 secure element protocol driver (SPI) devices"
depends on SPI && NXP_NFC_I2C
help
  This enables the Secure Element driver for SNxxx based devices.

  If unsure, say N.

  This selects Secure Element support.
```

7. Enable the below configuration in the platform config file to enable minimal FW download support from driver

```
NXP_NFC_RECOVERY=y
config NXP_NFC_RECOVERY
 bool "NXP based NFC minimal FW update support"
 depends on NXP_NFC_I2C && I2C
      default y
 help
   This enables NFC minimal FW update.

   If unsure, say N.
```

8. Set the kernel configuration in the platform config file, for example

```
CONFIG_ NXP_NFC_I2C=y
CONFIG_ NXP_ESE_P73=y
CONFIG_NXP_NFC_RECOVERY=y
```

9. Compile the kernel using corresponding cross compiler and copy the generated <platform>. dtb and Zimage file to the ANDROID_ROOT/device/vendor/platform-kernel

Note: It is recommended to apply the patches manually.

6.2 Setup of Android NFC for Hikey

6.2.1 Downloading Android source code

Use following command to get source code for Android-<x>.<y>:

```
repo init -u https://android.googlesource.com/platform/manifest -b android-<x>.<y>
repo sync -f
```

Note: x & y represents Android major & minor versions

For detailed steps to download Android source code refer Android website:

http://source.android.com/source/downloading.html

6.2.2 Building the source code

Use android build instructions from Android website for building android OS image:
http://source.android.com/source/building.html

Build name for Hikey development board is **hikey960.** For device specific build (e.g. Hikey), additional steps as described in link below needs to be followed.
https://source.android.com/setup/build/running

Information about the public APIs supported by Android NFC are available on following links:
http://developer.android.com/reference/android/nfc/package-summary.html

http://developer.android.com/reference/android/nfc/tech/package-summary.html

6.3 Android NXP NFC Android SW Delivery Package

6.3.1 Android NXP NFC Package Description

Project/Repository	Repository Link	Branch
NFC_NCIHAL_base	https://github.com/NXPNFCProject/NFC_NCIHAL_base	br_android_ncihalx_comm_12
NFC_NCIHAL_Nfc	*https://github.com/NXPNFCProject/NFC_NCIHAL_Nfc*	br_android_ncihalx_comm_12
NFC_NCIHAL_libnfc-nci	https://github.com/NXPNFCProject/NFC_NCIHAL_libnfc-nci	br_android_ncihalx_comm_12
nfcandroid_nfc_hidlimpl	https://github.com/NXPNFCProject/nfcandroid_nfc_hidlimpl	br_android_ncihalx_comm_12
nfcandroid_se_hidlimpl	https://github.com/NXPNFCProject/nfcandroid_se_hidlimpl	br_android_ncihalx_comm_12

nfcandroid_secureelement	https://github.com/NXPNFCProject/nfcandroid_secureelement	br_android_ncihalx_comm_12
nfcandroid_weaver_hidlimpl	https://github.com/NXPNFCProject/nfcandroid_weaver_hidlimpl	br_android_ncihalx_comm_12
nfcandroid_keymaster_hidlimpl	https://github.com/NXPNFCProject/nfcandroid_keymaster_hidlimpl	br_android_ncihalx_comm_12
nfcandroid_nxp_ese_clients	https://github.com/NXPNFCProject/nfcandroid_nxp_ese_clients	br_android_ncihalx_comm_12
NXPNFC_Reference	https://github.com/NXPNFCProject/NXPNFC_Reference	br_android_ncihalx_comm_12
NXPNFC_I2CDriver	https://github.com/NXPNFCProject/NXPNFC_I2CDriver	br_android_ncihalx_comm_12
NXPESE_SPIDriver	https://github.com/NXPNFCProject/NXPESE_SPIDriver	br_android_ncihalx_comm_12
NFC_NCIHAL_docs	https://github.com/NXPNFCProject/NFC_NCIHAL_docs	br_android_ncihalx_comm_12
nfc-NXPNFCC_FW	https://github.com/NXP/nfc-NXPNFCC_FW/tree/master/sn100x	Master
NXPAndroidDTA	https://github.com/NXPNFCProject/NXPAndroidDTA	Master
nfcandroid_frameworks	https://github.com/NXPNFCProject/nfcandroid_frameworks.git	br_android_ncihalx_comm_12

Table 1: Android NXP NFC Package Description

6.3.2 Integration of NXP NFC Modules

Modify/Add AOSP directories in-place with NXP GitHub sources as per the following table:

Module	NXP GitHub sources	Integration Path	Description
NFC Interface and Public APIs	NFC_NCIHAL_base /core/java/ android/nfc	$ANROID_ROOT/frameworks/base/c ore/ java/android/nfc	NFC Interfaces & Public APIs for Google AOSP
NFC JNI and JAVA implementation of NCI stack	*NFC_NCIHAL_Nfc* /nci	$ANDROID_ROOT/packages/apps/Nfc/nci	Includes Java files and JNI for NCI NFC stack. It is modified minimally to adapt new features provided by NXP.
	NFC_NCIHAL_Nfc /nci/jni/ extns/pn54x	#ANDROID_ROOT/packages/apps/Nfc/nci/jni/ extns/pn54x	It is an implementation of extension features developed by NXP. E.g. Mifare classic support
	NFC_NCIHAL_Nfc	$ANDROID_ROOT/packages/apps/Nfc [Remaining parts]	It is a derived module originally from AOSP. It is modified minimally to adapt new features provided by NXP.
NCI based NFC stack implementation	*NFC_NCIHAL_libnfc-nci*	$ANDROID_ROOT/system/nfc	NCI based NFC stack. It is a derived module originally from AOSP (Android Open Source Project).

			It is modified to adapt new features provided by NXP
HAL implementation for NFC	nfcandroid_nfc_hidlimpl	$ANDROID_ROOT/hardware/nxp/nfc	Hardware abstraction layer for NXP specific controllers. This directory includes the configuration files also as below. 1.libnfc-nci.conf (to be pushed to vendor/etc on target) 2.libnfc-nxp-sn100x_example.conf (to be pushed to vendor/etc on target as libnfc-nxp.conf. 3.libnfc-nxp_RF-sn100x_example.conf(to be pushed to /vendor/ on target) NOTE: these configuration files are example files. Contact NXP support engineer for creating exact file for your platform.
HAL implementation for Secure Element	nfcandroid_se_hidlimpl	$ANDROID_ROOT/hardware/nxp/secure_element	Hardware abstraction layer implementation for Secure Element.
HAL implementation for Weaver	nfcandroid_weaver_hidlimpl/weaver	$ANDROID_ROOT/hardware/nxp/weaver	Hardware abstraction layer implementation for Weaver.
HAL implementation for Secure Element	nfcandroid_keymaster_hidlimpl/keymaster	$ANDROID_ROOT/hardware/nxp/keymaster	Hardware abstraction layer implementation for Keymaster.
SE Service	nfcandroid_secureelement	$ANDROID_ROOT/packages/apps/SecureElement	AOSP Secure Element Service
eSe Client Library	nfcandroid_nxp_ese_clients	$ANDROID_ROOT/hardware/nxp/secure_element_extns	NXP eSE client library implementation
Vendor APIs	nfcandroid_frameworks	$ANDROID_ROOT/vendor/nxp/frameworks	NXP vendor framework APIs for NXP extension interfaces, SEMS & GSMA interfaces.
NFC I2C Driver	NXPNFC_I2CDriver/nfc	$KERNEL_ROOT/drivers/nxp/nfc	NFCC I2C Interface
NFC SPI Driver	NXPESE_SPIDriver/ese	$KERNEL_ROOT/drivers/nxp/ese	NFCC SPI Interface
Nxp Nfc Documentation	NFC_NCIHAL_docs	NA	NXP framework Java Docs
NFCC Firmware	nfc-NXPNFCC_FW	$ANDROID_ROOT/system/vendor/lib64	NFCC FW binary
DTA	NXPAndroidDTA	$ANDROID_ROOT/system/nfc-dta/	Device Test Application (DTA) used for NFC Forum testing.

Table 2 : Android NXP NFC Integration

6.3.3 Android NFC Apps and Lib on Target

Projects	Compiled Files	Location in target device
NFCNCIHAL_base/core/java/android/nfc	Will be part of framework.jar	/system/framework
NFC_NCIHAL_Nfc	lib/ NfcNci.apk oat/ libnfc_nci_jni.so	/system/app/NfcNci /system/lib64/
nfcandroid_secureelement	oat/ SecureElement.apk	/sytem/app/SecureElement
NFC_NCIHAL_libnfc-nci	libnfc_nci.so	/system/lib64
nfcandroid_nfc_hidlimpl	nfc_nci_nxp.so android.hardware.nfc@1.2-service	/vendor/lib64 /vendor/bin/hw/
nfcandroid_nfc_hidlimpl/extns	vendor.nxp.nxpnfc@2.0.so	/system/lib64
nfcandroid_se_hidlimpl	ese_spi_nxp.so android.hardware.secure_element@1.1-service android.hardware.secure_element@1.2-service	/vendor/lib64 /vendor/bin/hw/
nfcandroid_se_hidlimpl/extns	vendor.nxp.nxpese@1.0.so	/system/lib64
nfcandroid_keymaster_hidlimpl	libJavacardKeymaster41.so android.hardware.keymaster@4.1-javacard.service libese_transport	/vendor/lib64 /vendor/bin/hw /vendor/lib64
nfcandroid_weaver_hidlimpl	ese_weaver.so android.hardware.weaver@1.0-service	/vendor/lib64 /vendor/bin/hw
nfcandroid_nxp_ese_clients	se_extn_client.so ls_client.so jcos_client.so	/vendor/lib64
nfcandroid_hidlintf_nfc	android.hardware.nfc@1.0.so android.hardware.nfc@1.1.so android.hardware.nfc@1.2.so	/system/lib64
Nfcandroid_frameworks	com.gsma.services.nfc.jar com.nxp.nfc.jar com.nxp.sems.jar	/system/framework /vendor/framework

Table 3 : Android NXP NFC Apps & Library Info on Target

6.3.4 Android Platform Modifications

6.3.4.1 Android platform specific patches

Follow Step 1 & Step 2 to enable the following:

- Enable NFC, host card emulation and HCE-Felica features.
- Provide permission to i2c(pn553) and spi(p73) driver for NFC Hal and SE Hal
- Assign object type for i2c(pn553) and spi(p73) devices for providing se policy permissions
- Android SE Policy changes (these changes help in defining types, classes, permissions and rules for Nfc, SE and TEE Hal service)

1. Copy “nxp” folder in the below link to $ANDROID_ROOT/vendor

   https://github.com/NXPNFCProject/NXPNFC_Reference/tree/br_android_ncihalx_comm_12/nxp

2. Apply below patch in ANDROID_ROOT/device/linaro/hikey folder
   https://github.com/NXPNFCProject/NXPNFC_Reference/blob/br_android_ncihalx_comm_12/platform_patches/AROOT_device_linaro_hikey.patch

6.3.4.2 Android platform stability patches

Apply the below patch to avoid android platform stability issues like

https://github.com/NXPNFCProject/NXPNFC_Reference/blob/br_android_ncihalx_comm_12/platform_patches/AROOT_frameworks_base.patch

6.3.4.3 Android Source Build

To perform a full build, execute the following command from android root directory:

- cd $ANDROID_ROOT/
- make api-stubs-docs-update-current-api
- make system-api-stubs-docs-update-current-api
- make -j2

6.4 Host SW Source Package Compilation

6.4.1 Compilation Flags

NXP_EXTNS=TRUE	Enable NXP extensions

Table 4: Compilation Flags

6.4.2 Configuration Files

Host specific configuration are available in the below path and all the configs are self-explanatory and some of the configs are listed below

SN110 Config path:

https://github.com/NXPNFCProject/NXPNFC_Reference/tree/br_android_ncihalx_comm_12/nxp/SNxxx/hw/SN1xx/sn110.

SN100 config path:

https://github.com/NXPNFCProject/NXPNFC_Reference/tree/br_android_ncihalx_comm_12/nxp/SNxxx/hw/SN1xx/sn100

6.4.2.1 Configuration in libnfc-*.conf file

Configuration	Description
NFC_DEBUG_ENABLED	Application option to enable and disable logs. Enable 0x01 Disable 0x00
UICC_LISTEN_TECH_MASK	Force UICC to only listen to a specific technology. By default, the value is 0x07 i.e. it listens for the following technologies: Tech A, B and F
POLLING_TECH_MASK	Force Tag polling for the different Tech Notable bits: NFC Technology A 0x00 NFC Technology B 0x02 NFC Technology F 0x04 Proprietary Technology ISO15693 0x08 NFC Technology A active mode 0x40 NFC Technology F active mode 0x80 Default = 0x6F (A|B|F|ISO15693|B_PRIME|KIVIO|A_ACTIVE|F_ACTIVE)
P2P_LISTEN_TECH_MASK	Force P2P to only listen for the following technology(s) Notable bits: NFC Technology A 0x00 NFC Technology F 0x04 NFC Technology A active mode 0x40 NFC Technology F active mode 0x80 Default = 0x6F (A|F|A_ACTIVE|F_ACTIVE)
PRESERVE_STORAGE	0x01 Preserves *.bin files 0x00 Deletes *.bin files
NFA_MAX_EE_SUPPORTED	Override the stack default for FA_EE_MAX_EE_SUPPORTED. The value is set to 3 by default as it assumes we will discover 0xF2, 0xF3, and 0xF4. If a platform will exclude and SE, this value can be reduced

	so that the stack will not wait any longer than necessary.
NFA_DM_DISC_NTF_TIMEOUT	Deactivate notification wait time out in seconds used in ETSI Reader mode. 0x00 Infinite wait
AID_MATCHING_MODE	Defines how the AID should be matched AID_MATCHING constants AID_MATCHING_EXACT_ONLY 0x00 AID_MATCHING_EXACT_OR_PREFIX 0x01 AID_MATCHING_PREFIX_ONLY 0x02 AID_MATCHING_EXACT_OR_SUBSET_OR_PREFIX 0x03
NFA_AID_BLOCK_ROUTE	0x00 Disable Black list 0x01 Enable Black list
NXP_WM_MAX_WTX_COUNT	Maximum WTX requests entertained by MW
DEFAULT_SYS_CODE	Set the default Felica T3T System Code: These settings will be used when application does not set this parameter
NXP_NFC_DEV_NODE	Nfc Device Node name i.e. "/dev/pn553"
MIFARE_READER_ENABLE	Extension for Mifare reader enable. Default=0x01
LEGACY_MIFARE_READER	Configuration to enable or disable legacy Mifare Reader implementation 0: General implementation 1: Legacy implementation
NXP_FW_NAME	File name for Firmware i.e. "libsn100u_fw.so"
NXP_AUTONOMOUS_ENABLE	0x01 Enable Autonomous mode 0x00 Disable Autonomous mode
NXP_GUARD_TIMER_VALUE	Gurad Timer range to 0x0F-0xFF(15 to 255 seconds)
NXP_SYS_CLK_SRC_SEL	System clock source selection configuration 0x00 CLK_SRC_XTAL 0x01 CLK_SRC_PLL
NXP_SYS_CLK_FREQ_SEL	System clock frequency selection configuration CLK_FREQ_13MHZ 1 CLK_FREQ_19_2MHZ 2 CLK_FREQ_24MHZ 3 CLK_FREQ_26MHZ 4 CLK_FREQ_38_4MHZ 5 CLK_FREQ_52MHZ 6

NXP_DEFAULT_UICC2_SELECT	This is used to configure UICC2 at boot time. 0x03 UICC2
NXP_SWP_RD_TAG_OP_TIMEOUT	This configuration would be used to inform apps about secure reader timeout event when no tag tapped to reader within the configured timeout
DEFAULT_AID_ROUTE	Configuration to set default AID route. This settings will be used when application does not set this parameter host 0x00 eSE 0x01 UICC 0x02
DEFAULT_ISODEP_ROUTE	Set the ISODEP (Mifare Desfire) route Location : #This settings will be used when application does not set this parameter
DEFAULT_MIFARE_CLT_ROUTE	Configuration to set default mifare clt route location. host 0x00 eSE 0x01 UICC 0x02 UICC2 0x03
DEFAULT_FELICA_CLT_ROUTE	Configuration to set default mifare clt route location. host 0x00 eSE 0x01 UICC 0x02 UICC2 0x03
DEFAULT_SYS_CODE_ROUTE	Set the default Felica T3T System Code route Location host 0x00 eSE 0x01 UICC 0x02 UICC2 0x03
DEFAULT_AID_PWR_STATE	This settings will be used to configure power state for empty AID route entry in routing table bit pos 0 = Switch On bit pos 1 = Switch Off bit pos 2 = Battery Off bit pos 3 = Screen off unlock bit pos 4 = Screen On lock bit pos 5 = Screen Off lock
DEFAULT_DESFIRE_PWR_STATE	This settings will be used to configure power state for ISO_DEP proto route entry in routing table

	bit pos 0 = Switch On bit pos 1 = Switch Off bit pos 2 = Battery Off bit pos 3 = Screen off unlock bit pos 4 = Screen On lock bit pos 5 = Screen Off lock
DEFAULT_MIFARE_CLT_PWR_STATE	This settings will be used to configure power state for Type A & B tech route entry in routing table bit pos 0 = Switch On bit pos 1 = Switch Off bit pos 2 = Battery Off bit pos 3 = Screen off unlock bit pos 4 = Screen On lock bit pos 5 = Screen Off lock
DEFAULT_FELICA_CLT_PWR_STATE	This settings will be used to configure power state for Felica tech route entry in routing table bit pos 0 = Switch On bit pos 1 = Switch Off bit pos 2 = Battery Off bit pos 3 = Screen off unlock bit pos 4 = Screen On lock bit pos 5 = Screen Off lock
DEFAULT_T4TNFCEE_AID_POWER_STATE	This settings will be used to configure power state for T4T NFCEE NDEF AID entry in routing table bit pos 0 = Switch On bit pos 1 = Switch Off bit pos 2 = Battery Off bit pos 3 = Screen off unlock bit pos 4 = Screen On lock bit pos 5 = Screen Off lock
NXP_P61_LS_DEFAULT_INTERFACE	P61 Loader Service interface options NFC 0x01 SPI 0x02
NXP_P61_JCOP_DEFAULT_INTERFACE	P61 interface options for JCOP Download NFC 0x01 SPI 0x02
NFA_POLL_BAIL_OUT_MODE	Bail out mode

	If set to 1, NFCC is using bail out mode for either Type A or Type B poll. Default value is 0x01
PRESENCE_CHECK_ALGORITHM	Choose the presence-check algorithm for type-4 tag. If not defined, the default value is 1. #0 NFA_RW_PRES_CHK_DEFAULT; Let stack selects an algorithm # 1 NFA_RW_PRES_CHK_I_BLOCK; ISO-DEP protocol's empty I-block # 2 NFA_RW_PRES_CHK_ISO_DEP_NAK; Type - 4 tag protocol iso-dep nak presence check command is sent waiting for rsp and ntf.
NFA_PROPRIETARY_CFG	It will be used to specify vendor Proprietary Protocol & Discovery Configuration Set to 0xFF if unsupported # byte[0] NCI_PROTOCOL_18092_ACTIVE # byte[1] NCI_PROTOCOL_B_PRIME # byte[2] NCI_PROTOCOL_DUAL # byte[3] NCI_PROTOCOL_15693 # byte[4] NCI_PROTOCOL_KOVIO # byte[5] NCI_PROTOCOL_MIFARE # byte[6] NCI_DISCOVERY_TYPE_POLL_KOVIO # byte[7] NCI_DISCOVERY_TYPE_POLL_B_PRIME # byte[8] NCI_DISCOVERY_TYPE_LISTEN_B_PRIME
NXP_NCI_PARSER_LIBRARY	0x01 Enable Lx debug information 0x00 Disable
NXP_CORE_PROP_SYSTEM_DEBUG	This config will enable different level of Rf transaction debugs based on the following values provided. Decoded information will be printed in adb logcat Debug Mode Levels Disable Debug 0x00 L1 Debug 0x01 L2 Debug 0x02 L1 & L2 Debug 0x03 L1 & L2 & RSSI 0x04 L1 & L2 & Felica 0x05
HOST_LISTEN_TECH_MASK	Enable/Disable HOST to listen for a selected protocol # 0x00 : Disable Host Listen # 0x01 : Enable Host to Listen (A) for ISO-DEP tech A # 0x02 : Enable Host to Listen (B) for ISO-DEP tech B

	# 0x04 : Enable Host to Listen (F) for T3T Tag Type Protocol tech F # 0x07 : Enable Host to Listen (ABF)for ISO-DEP tech AB & T3T Tag Type Protocol tech F
FORWARD_FUNCTIONALITY_ENABLE	This config will be used to enable or disable the card emulation support for offshost SE's which are either type A or type B only # Disable 0x00 # Enable 0x01
OFF_HOST_ESE_PIPE_ID=0x16 OFF_HOST_SIM_PIPE_ID=0x0A OFF_HOST_SIM2_PIPE_ID=0x23	Configure the NFC Extras to open and use a static pipe. If the value is not set or set to 0, then the default is use a dynamic pipe based on a destination gate (see NFA_HCI_DEFAULT_DEST_GATE). Note there is a value for each EE (ESE/SIM1/SIM2)
NXP_FLASH_CONFIG	Flashing Options Configurations FLASH_UPPER_VERSION 0x01 FLASH_DIFFERENT_VERSION 0x02 FLASH_ALWAYS 0x03

Table 5: Configuration Flags

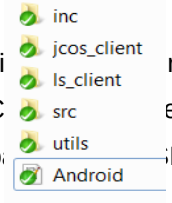
6.5 Feature Integration guideline

6.5.1 eSE client service integration

Implementation for clients is present in nfcandroid_nxp_ese_clients project. Project can be downloaded from the below link

https://github.com/NXPNFCProject/nfcandroid_nxp_ese_clients

Below is the folder structure.



Uti rce present in **utils**, **src** and **inc** folders

JC e library is present in **jcos_client** folder

Lo EMS agent implementation is present in **ls_client** folder

Below are the steps to integrate eSE clients (JCOP OS update library, loader service/ SEMS agent):

- Copy nxp_ese_clients folder as hardware/nxp/secure_element_extns

```
cp -rf nxp_ese_cleints $ANDROID_ROOT/hardware/nxp/secure_elements_extns
```

To enable JCOP download during boot time below are the preconditions

1. Push update scripts and apdu file to predefined location
    a. APDU files, jci.jcsh & cci.jcsh scripts shall be present at "$ANDROID_ROOT/vendor/etc/".
2. Enable JCOP OS download interface options as below.

a. P61 interface parameter NXP_P61_JCOP_DEFAULT_INTERFACE can be set to the following for JCOP Download

NFC 0x01

SPI 0x02

NXP_P61_JCOP_DEFAULT_INTERFACE=0x02 or 0x01

3. Optionally enable NXP_JCOP_FORCE_UPDATE_REQUIRED=0x01 only when force update required during normal boot (other than very first boot or factory update)

To perform LS/SEMS script update during boot time below are preconditions:

1. Push loaderservice_updater.txt to vendor/etc/ folder in android filesystem.
2. Enable LS/SEMS update interface option as below

NFC 0x01

SPI 0x02

NXP_P61_LS_DEFAULT_INTERFACE=0x01 or 0x02

3. Optionally enable below flag to perform script update for every boot (other than very first boot or after factory update)

Option to perform LS update every boot

Enable 0x01

Disable 0x00

NXP_LS_FORCE_UPDATE_REQUIRED=0x01

6.5.2 OMAPI Secure Element terminal configuration

Assignment of terminal number to each SE interface (SMB, SPI-REE and SPI-TEE) is based on system configuration in **libnfc-nxp-SN100-example.conf**. These terminals are mapped to OMAPI framework SEService readers list.

Terminal Naming should start from eSE1 and continue in ascending order

(This is as per OMAPI SE service implementation)

Only terminal which are mapped in configuration file are reflected as readers available in SE service.

For Example: -

Configuration1 (Both terminals available eSE1 and eSE2)

Order below is just an example

NXP_SPI_SE_TERMINAL_NUM="**eSE1**" -> eSE domain accessed via SPI interface

NXP_NFC_SE_TERMINAL_NUM="**eSE2**" -> eSE-REE domain accessed via SMB/DWP interface

Configuration2(Any one terminal available eSE1)

Option1

NXP_SPI_SE_TERMINAL_NUM="eSE1" -> eSE domain accessed via SPI interface

Option2

NXP_NFC_SE_TERMINAL_NUM ="eSE1" -> eSE-REE domain accessed via SPI interface

Additionally, from Android 11 onwards it is mandatory to enable terminals as per the system configuration in vendor/etc/vintf/manifest.xml

Based on number of terminals getting enabled in config file corresponding number of terminal instances need to be updated in manifest.xml as shown below

```
<hal format="hidl">
    <name>android.hardware.secure_element</name>
    <transport>hwbinder</transport>
    <version>1.1</version>
    <interface>
        <name>ISecureElement</name>
        <instance>eSE1</instance>
        <instance>eSE2</instance>                :
            :
    </interface>
    <fqname>@1.1::ISecureElement/eSE1</fqname>
    <fqname>@1.1::ISecureElement/eSE2</fqname>
</hal>
```

6.5.3 NFC DTA Setup

6.5.3.1 NFC DTA Source

Information of NXPAndroidDTA Project repositories in the GitHub are as below:

NFC DTA source can be downloaded from the below link:

https://github.com/NXPNFCProject/NXPAndroidDTA

Copy NFC DTA source to /system/nfc-dta/ folder

6.5.3.2 Build NFC DTA

After building DTA, it generates 64-bit DTA binaries. To install DTA on the android device, ensure that adb is installed on the system and USB cable is connected between the system and the android device.

6.5.3.3 NFC DTA Binaries

1. The generated binary files should be pushed to the target devices as per the below table.

Project	Compiled Files	Location in target device
/system/nfc-dta/	libdta.so libosal.so libdta_jni.so libmwif.so	/system/lib64
/system/nfc-dta/	NxpDTA.apk	/system/app/NxpDTA (Create folder “NxpDTA” under /system/app in target device)

After updating the required files, the “NXP Device Test Application” appears in the main menu.

Setting to be done before running DTA APK are as below

1. Switch off the default NFC service option in Settings.

   Settings->Connected Devices >NFC as OFF (Un-ticked) and reboot the device (using ‘adb reboot’).
2. Set Screen time out settings or Stay Awake option should be ticked.

   Screen time out should be updated in the IUT settings to avoid the DTA RF signal loss. Because once the device goes to sleep mode immediately RF will be stopped from device, to avoid this device screen timeout should be increased to 30 minutes or device should powered. The following path can be used for updating the screen timeout setting.

   Main menu -> Settings -> Developer Options -> Stay Awake.

   Settings -> Display -> Sleep -> select 30 minutes.

6.5.4 Firmware Download

NXP provides precompiled firmware for ARM platforms. NXP also can provide firmware as .c file and it can be compiled as .so file with the platform compiler. Firmware resides at location `/system/vendor/lib64/` on the android target system. The firmware filename can be set in NXP_FW_NAME configuration in libnfc-nxp.conf file

Firmware can be updated when NXP releases an updated version. Steps to update are as follows:

1. Compile the firmware to .so file using the file received in .C file format. If firmware is in .so format then this step can be skipped.
2. Set the FW name in libnfc-nxp.conf file in NXP_FW_NAME
3. Push the firmware file to `/system/vendor/lib64` directory on target.
4. Reboot the device or disable and enable NFC service. New firmware will be downloaded during the NFC service boot up
5. Firmware file can be downloaded from below location

   https://github.com/NXP/nfc-NXPNFCC_FW/tree/master/sn1xx

Note 1: Firmware download can take up to 10 seconds including host delay.
Note 2: It is strongly recommended not to modify the original firmware download logic of Android NFC.

Note 3: It is recommended that Firmware is always upgraded and not downgraded. If firmware version is required to be downgraded, then please consult NXP.

6.6 Android one specific

Android one compliant stack is where only vendor partition(HAL source), config files are from NXP remaining layers(Framework, NFC service, JNI and libnfc source) i.e. system partition is default AOSP source

6.6.1 Card emulation through Off-host in Android-one platform

To achieve card emulation functionality through off-host(eSE/UICC) on Android one stack

below changes are needed in libnfc-nxp config file which is different from regular config options

Default AOSP implementation only supports below config options related to routing table management

1) DEFAULT_ISODEP_ROUTE(libnfc-nci.conf)
2) DEFAULT_SYS_CODE_ROUTE(libnfc-nxp.conf)
3) DEFAULT_OFFHOST_ROUTE(libnfc-nxp.conf)

	Value	
Route	**Android One**	**Regular**
eSE	0xC0	0x01
UICC1	0x80	0x02
UICC2	0x81	0x03

Hence the platforms which are willing to use Card emulation functionality through off-host locations shall update config file with values indicated above

7. Legal information

7.1 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.

7.2 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.

7.3 Licenses

Purchase of NXP <xxx> components
<License statement text>

7.4 Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

<Patent ID> — owned by <Company name>

7.5 Trademarks

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

<Name> — is a trademark of NXP Semiconductors N.V.

Table of Contents