AN11762

Android NFC Setup Guide

Rev. 2.7 — 1 July 2021 Application note

Document information

Info	Content
Keywords	NFC, Android

Revision history

Rev	Date	Description
1.0	2015-11-06	Initial version for NXP NFC Setup Guide
1.1	2016-01-14	Minor review updates related to compilation flags and configurations
1.2	2016-05-27	HOST_LISTEN_ENABLE is replaced by HOST_LISTEN_TECH_MASK
1.3	2016-08-01	Added references for PN553/PN80T
1.4	2016-11-18	Added CR9 DTA screen shots.
1.5	2017-09-13	Updates with respect to Android Oreo included Added CR9/ CR10 DTA screenshots along with necessary description.
1.6	2018-03-14	Editorial changes & added Connection Device Limit zero DTA UI in the DTA APK.
1.7	2018-05-15	Updates with respect to OMAPI support in Android
1.8	2018-06-21	Android P migration OMAPI service integration
1.9	2018-08-24	Minor correction in path names (Section 5.4)
2.0	2019-03-29	Updated guidelines for JCOP-OSU, LS, GSMA and dynamic configurations
2.1	2019-05-29	Android 10 migration Updates on LS at run time. Multi-SE & Driver cleanup
2.2	2019-07-12	Filed Detect feature.
2.3	2019-09-03	Minor correction in Section 7 and 13.2
2.4	2019-11-25	T4T NDEF emulation from eSE.
2.5	2020-05-09	Initial version for Android 11 NXP NFC Setup Guide
2.6	2020-08-09	Android-11 Beta release alignment and minor bug fix’s
2.7	2021-07-01	Android-12 Beta 2 release alignment and minor bug fix’s

Contact information

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

1. Introduction

NXP’s NCI based NFC controllers (PN553/PN80T/PN557/PN81T/NQ330/NQ440) are designed to work with Android open source using NCI based stack for Android NFC. In the further sections, NXP-NFCC refers to PN553/PN80T/PN557/PN81T/NQ330/NQ440.

2. Scope

This setup guide provides guidelines for setting up NXP NFC in android build environment. It is an example guideline for basic system integration. OEM integration may have variations based on actual system and integration.

3. 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

Picture below shows basic flow for Android NFC SW bring up. Following sections describe these steps in detail.

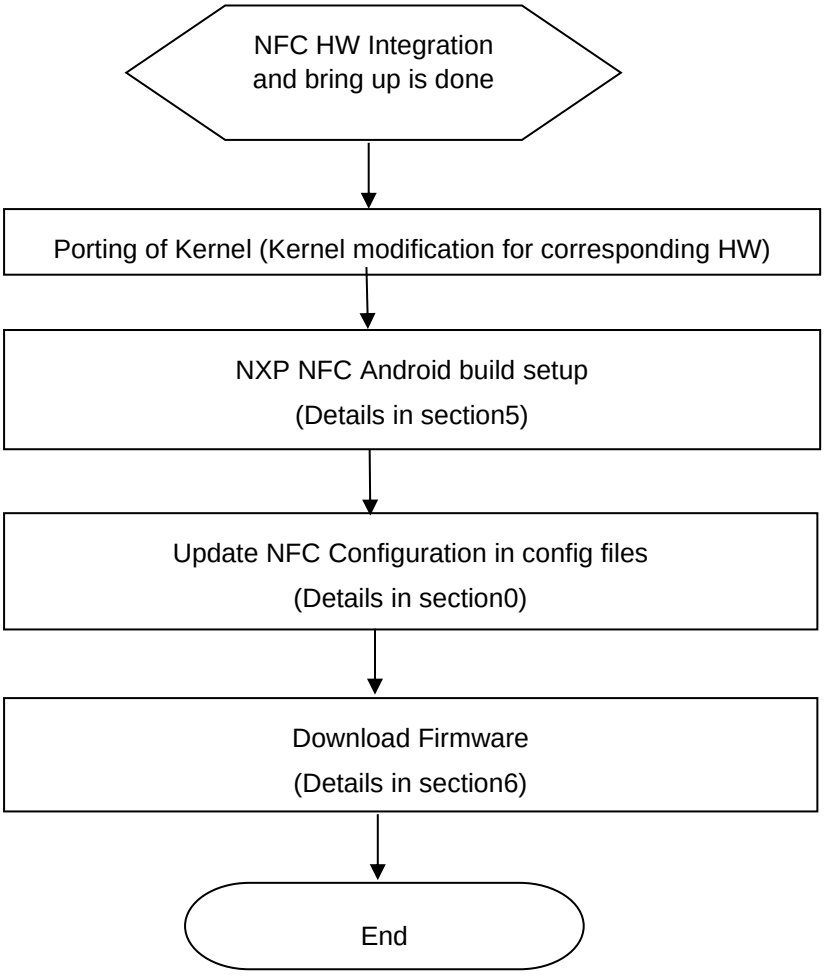


Figure 1: Android NFC SW bring up flow

4. Architecture Overview

Figure 2 describes the architecture of Android NFC with NXP-NFCC. NCI HALx provides the Hardware abstraction for the NXP's NFCC. NXP additional features/enhancements are part of NXP Extensions provided on top of AOSP NFC Stack.

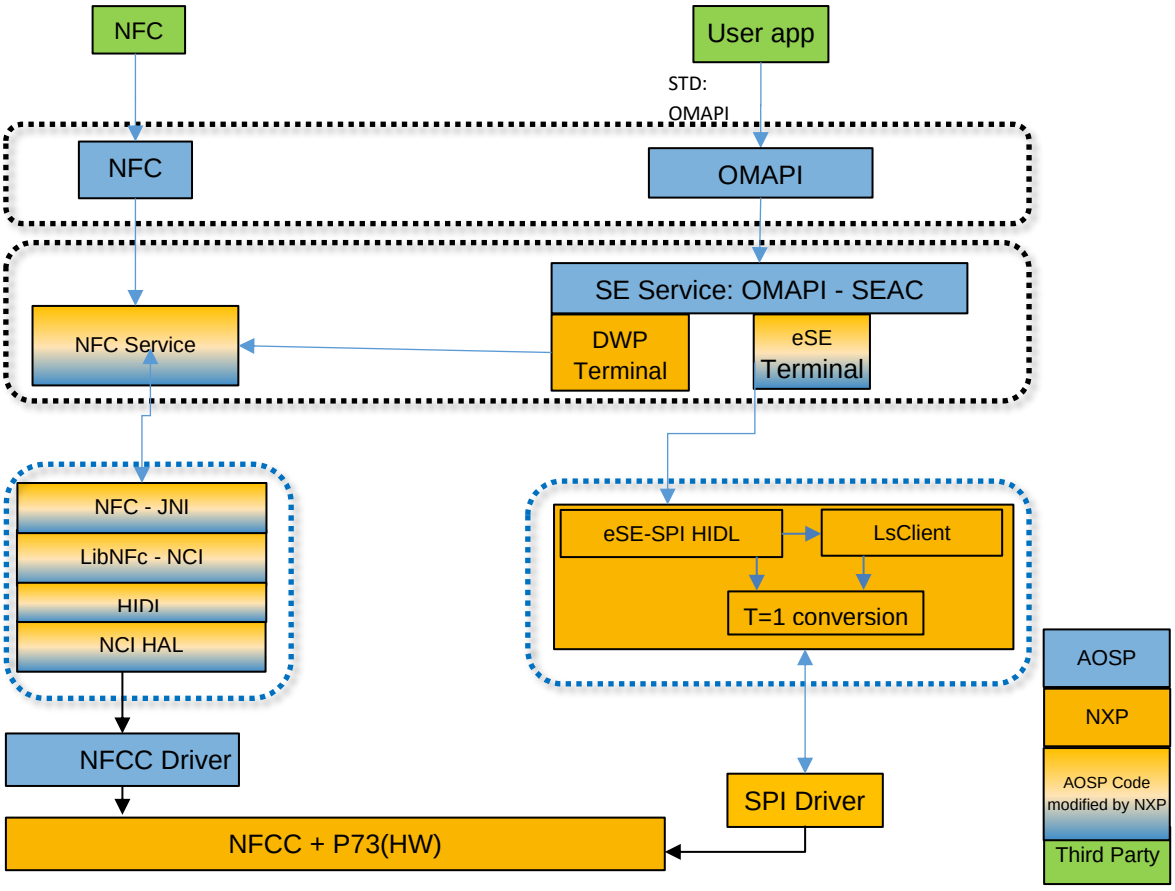


Figure 2: Android NFC with NXP-NFCC

5. Setup of Android NFC

5.1 Downloading Android Source Code

Following the instructions from Android website:

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

Use following command to get source code for respective branch Android-x.y:

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

5.2 Building the Android Source Code

Use android build instructions from Android website for building android OS image:

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

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

5.3 Android NFC package description

Information of NXP's NFC Project repositories in the GitHub are as below:

Repository Name	Checkout Command
NFC_NCIHAL_Nfc	*git clone https://github.com/NXPNFCProject/NFC_NCIHAL_Nfc.git* The contents of this folder needs to be placed in packages/apps/Nfc directory in the android build.
NFC_NCIHAL_libnfc-nci	*git clone https://github.com/NXPNFCProject/NFC_NCIHAL_libnfc-nci.git* The contents of this folder needs to be placed in system/nfc directory in the android build.
NFC_NCIHAL_base	*git clone https://github.com/NXPNFCProject/NFCNCIHAL_base.git* The contents of this folder needs to be merged in frameworks/base directory in the android build.
nfcandroid_nfc_hidlimpl	*git clone https://github.com/NXPNFCProject/nfcandroid_nfc_hidlimpl.git* The content of this folder needs to be merged in hardware/nxp/nfc in the android build.
nfcandroid_se_hidlimpl	git clone https://github.com/NXPNFCProject/nfcandroid_se_hidlimpl.git The content of this folder needs to be merged in hardware/nxp/secure_element in the android build.

nfcandroid_nxp_ese_clients	git clone https://github.com/NXPNFCProject/nfcandroid_nxp_ese_clients.git The content of this folder needs to be merged in hardware/nxp/secure_element_extns in the android build.
nfcandroid_secureelement	*git clone https://github.com/NXPNFCProject/nfcandroid_secureelement.git* The content of this folder needs to be merged in Packages/apps/SecureElement in the android build.
nfcandroid_frameworks	*git clone https://github.com/NXPNFCProject/nfcandroid_frameworks.git* *The content of this folder needs to be merged in vendor/nxp in the android build*
NXPNFC_Reference	git clone https://github.com/NXPNFCProject/NXPNFC_Reference.git Patches inside this repo to be aligned to respective build directories.
NXPNFCC_FW	*git clone https://github.com/NXP/nfc-NXPNFCC_FW.git*

Description of the contents of the directories of NXP's NFC Project in the GitHub are as below:

Module Type	Path	Description
NFC Interfaces and public APIs	NFC_NCIHAL_base/core/java/android/nfc NFC_NCIHAL_base/core/java/android/se	Contains Android NFC Framework files. Contains OMAPI framework interfaces
	nfcandroid_frameworks	NXP's extensions to android NFC Framework
NFC JNI and Java implementation of NCI stack	NFC_NCIHAL_Nfc/nci	Contains files for Nfc Nci stack.
	NFC_NCIHAL_Nfc/nci/jni/extns/pn54x	Contains implementation of extension features developed by NXP in JNI layer. E.g. Mifare classic support.
	NFC_NCIHAL_Nfc [Remaining parts]	Contains Android NFC application source files.
NCI based NFC stack implementation	NFC_NCIHAL_libnfc-nci	Contains NCI based Native NFC stack
HAL implementation	nfcandroid_nfc_hidlimpl	Contains hardware abstraction layer for NXP specific controllers
	nfcandroid_se_hidlimpl	Contains T=1 protocol stack and hardware abstraction layer for eSE
	nxp_ese_clients	Contains JCOP Download interface for eSE.
Firmware	NXPNFCC_FW	It is a directory, which includes the firmware file for NXP-NFCC

5.4 Integration of NXP NFC and Secure Element Modules

5.4.1 Modify AOSP directories in-place with NXP GitHub sources

Create folder to download repositories from GitHub

- cd $ANDROID_ROOT
- mkdir NXPNFCProject
- export NXP_GIT=$ANDROID_ROOT/NXPNFCProject

Download the following subprojects for the branch br_android_ncihalx_row_12 and extract in folder $NXP_GIT

- NFC_NCIHAL_base
- NFC_NCIHAL_libnfc-nci
- NFC_NCIHAL_Nfc
- NXPNFC_Reference
- nfcandroid_nfc_hidlimpl
- nfcandroid_se_hidlimpl
- nfcandroid_nxp_ese_clients
- nfcandroid_secureelement
- nfcandroid_frameworks
- NXPESE_SPIDriver
- NXPNFC_I2CDriver

Download NXPAndroidDTA and extract in folder $NXP_GIT

Replace content of $ANDROID_ROOT/frameworks/base/core/java/android/nfc by content of NFC_NCIHAL_base/core/java/android/nfc

- cd $ANDROID_ROOT/frameworks/base
- rm -rf core/java/android/nfc
- cp -rf $NXP_GIT/NFC_NCIHAL_base/core/java/android/nfc core/java/android/

Replace content of $ANDROID_ROOT/frameworks/base/core/java/android/se by content of NFC_NCIHAL_base /core/java/android/se

- cd $ANDROID_ROOT/frameworks/base
- rm -rf core/java/android/se
- cp -rf $NXP_GIT/NFC_NCIHAL_base/core/java/android/se core/java/android/

Update $ANDROID_ROOT/core/res/res/values/attrs.xml for ApduPatternGroup changes from $NXP_GIT/NXPNFC_Reference/platform_patches/AROOT_frameworks_base.patch

Replace content of $ANDROID_ROOT/system/nfc by content of NFC_NCIHAL_libnfc-nci

- cd $ANDROID_ROOT/system/nfc
- rm -rf *
- cp -rf $NXP_GIT/NFC_NCIHAL_libnfc-nci/* .

Replace content of $ANDROID_ROOT/packages/apps/Nfc by content of NFC_NCIHAL_Nfc

- cd $ANDROID_ROOT/packages/apps/Nfc
- rm -rf *
- cp -rf $NXP_GIT/NFC_NCIHAL_nfc/* .

Copy folder nfcandroid_frameworks into folder $ANDROID_ROOT/vendor/nxp/framework

- cd $ANDROID_ROOT/vendor/nxp/framework
- cp -rf $NXP_GIT/nfcandroid_frameworks /vendor/nxp/framework

Replace content of $ANDROID_ROOT/packages/apps/SecureElement by content of nfcandroid_secureelement

- cd $ANDROID_ROOT/packages/apps/SecureElement
- rm -rf *
- cp -rf $NXP_GIT/ nfcandroid_secureelement/* .

Replace content of $ANDROID_ROOT/hardware/interfaces/nfc by content of nfcandroid_hidlintf_nfc

- cd $ANDROID_ROOT/hardware/interfaces/nfc
- rm -rf *
- cp -rf $NXP_GIT/nfcandroid_hidlintf_nfc/* .

Replace content of $ANDROID_ROOT/hardware/nxp/ secure_element by content of nfcandroid_se_hidlimpl

- cd $ANDROID_ROOT/hardware/nxp/secure_element
- rm -rf *

- cp -rf $NXP_GIT/ nfcandroid_se_hidlimpl/* .

Create folder $ANDROID_ROOT//hardware/nxp/secure_element_extns and copy the content of nfcandroid_nxp_ese_clients

cd $ANDROID_ROOT/system

- mkdir secure_element_extns
- cd secure_element_extns
- cp -rf $NXP_GIT/ secure_element_extns /* .

Create folder $ANDROID_ROOT/system/nfc-dta and copy the content of NXPAndroidDTA

cd $ANDROID_ROOT/system

- mkdir nfc-dta
- cd nfc-dta
- cp -rf $NXP_GIT/NXPAndroidDTA/nfc-dta/* .

5.4.2 Build the full Android image

Execute the following commands:

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

If required, the individual NXP NFC modules can be built by executing the following commands:

- cd $ANDROID_ROOT/frameworks/base
- mm
- cd $ANDROID_ROOT/vendor/nxp
- mm
- cd $ANDROID_ROOT/hardware/nxp/nfc
- mma
- cd $ANDROID_ROOT/hardware/nxp/secure_element
- mma
- cd $ANDROID_ROOT/hardware/nxp/secure_element_extns
- mma
- cd $ANDROID_ROOT/system/nfc

- mma
- cd $ANDROID_ROOT/packages/apps/SecureElement
- mma
- cd $ANDROID_ROOT/packages/apps/Nfc
- mma

5.5 Android NFC Apps and Lib on Target

Projects	Compiled Files	Location in target device
NFCNCIHAL_base/core/java/android/nfc NFCNCIHAL_base/core/java/android/se	Will be part of framework.jar	/system/framework
nfcandroid_frameworks	com.nxp.nfc.jar	/system/framework
NFC_NCIHAL_Nfc	lib/ NfcNci.apk oat/	/system/app/NfcNci
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	/system/vendor/lib64 system/vendor/bin/hw/
nfcandroid_nfc_hidlimpl/extns.	vendor.nxp.nxpnfc@2.0.so vendor.nxp.nxpnfclegacy@1.0.so	/system/lib64
nfcandroid_se_hidlimpl.	ese_spi_nxp.so ls_client.so android.hardware.secure_element@1.2-service	/vendor/lib64 /vendor/lib64 /vendor/bin/hw/
nfcandroid_se_hidlimpl/extns.	vendor.nxp.nxpese@1.0.so	/system/lib64
nfcandroid_nxp_ese_clients.	ese_client.so	/vendor/lib64

Following table lists the binaries used for Android NFC using NCI based stack.

5.6 Building the Kernel Source Code

Follow the following steps for building the kernel:

- Create nxp folder inside kernel/driver/
- Copy nfc directory from NXPNFC_I2CDriver to kernel/drivers/nxp
- Copy ese directory from NXPNFC_SPIDriver to kernel/drivers/ese
- Compile the kernel using corresponding cross compiler and copy the generated <platform>. dtb and Zimage file to the ANDROID_ROOT/device/vendor/platform-kernel.

Generate the corresponding boot and dt (device tree) images for the target platform

5.7 References

Information about OMAPI APIs are available on below link:

https://developer.android.com/reference/android/se/omapi/package-summary.html

https://simalliance.org/handset/handset-technical-releases (legacy OMAPI package)

Information about ARA rules can be found in document Secure Element Access Control v1.1 available on below link:

https://globalplatform.org/specs-library

SEPolicy Information about SEPolicy rules are in the below link:
https://source.android.com/security/selinux/

Configuration files

There are configuration files used by Android NFC which is located in `system/etc` (`vendor/etc` for Android master) directory of target. These files are provided in GitHub project `nfcandroid_nfc_hidlimpl/halimpl/pn54x` as example files. This section describes the different flags in configuration files as examples. There are many additional flags which are explained in the config files as per NFC chip (please refer the config files part of GitHub releases).

5.8 Configurations in libnfc-nci.conf files

Configurations	Descriptions
`HOST_LISTEN_TECH_MASK`	Forcing HOST to listen for a selected technology • 0x00: Disable Host Listen • 0x01: Enable Host Listen for Tech A • 0x02: Enable Host Listen for Tech B • 0x03: Enable Host Listen for Tech AB
`NXP_FWD_FUNCTIONALITY_ENABLE`	In case a communication is initiated in a RF technology (A or B) supported by host or eSE,but not supported by UICC, the forward function allows to forward the ISO/IEC 14443 level 4 commands to UICC. • To Disable: Set to 0x00 • To Enable: Set to 0x01 • Default: 0x01

5.9 Configurations in libnfc-nxp.conf file

Configurations	Descriptions
DEFAULT_AID_ROUTE	Configuration to set default route location for AID. This settings will be used when application does not set this parameter using the DefaultRouteSet() API defined in android framework. • Host: 0x00 • eSE (embedded Secure Element): 0x01 • UICC: 0x02 • Default: 0x00 However, if the NFCC routing table entries overflow with set default AID route, then the default routing location may be modified internally to accommodate all the AID's.
DEFAULT_ISODEP_ROUTE	Configuration to set default route location for ISO-DEP Protocol. • Host: 0x00 • eSE: 0x01 • UICC: 0x02 • Default: 0x02
DEFAULT_OFFHOST_ROUTE	Configuration to set default route location for AID's registered from application. This configuration's will be used only when the route location is not defined from the application. • eSE: 0x01 • UICC: 0x02 • Default: 0x01
DEFAULT_TECH_ABF_ROUTE	Configuration to set default route location for ABF Technology. • eSE: 0x01 • UICC: 0x02 • Default: 0x01
NXP_FW_NAME	Name of the firmware file (ex: libpn553_fw .so). This name shall be name of the file as in /vendor/lib64/ directory

5.10 Dynamic Configuration for NFC

NFC configurations can be updated at run time by setting the system property `adb shell setprop persist.nfc_cfg.config_file_name` to `"libnfc-nci.conf"` and `adb shell setprop persist.vendor.nfc.config_file_name file_name` to `"libnfc-nxp.conf"`. New Config files shall reside at location `/etc/libnfc-nci_configx.conf` and `/vendor/etc/libnfc-nxp_configx.conf` on target. Next NFC OFF to ON shall update configs as per the selected config file(`"configx"`). If the selected file is not present in target directory, then system shall load default configuration from `/product/etc/libnfc-nci.conf` & `/vendor/etc/libnfc-nxp.conf`.

Example to update the dynamic configuration for NFC:

1. Push the `libnfc-nci.conf` file with modified configuration to `/product/etc/libnfc-nci_config1.conf`.
2. Push the `libnfc-nxp.conf` file with modified configuration to `/vendor/etc/libnfc-nxp_config1.conf`.
3. Set the system property `"adb shell setprop persist.vendor.nfc.config_file_name "libnfc-nxp_config1.conf"`. For libnfc-nxp.conf file.
4. Set the system property `"adb shell setprop persist.nfc_cfg.config_file_name " libnfc-nci_config1.conf`. For libnfc-nci.conf file.
5. NFC OFF and ON to update the modified configuration.

Note1: There are example libnfc-nxp.conf files provided with release package. Please contact NXP support engineer to create the libnfc-nxp.conf for customer platform based on requirement and antenna design, which can be used for end product in production.

Note2: Optionally, NXP can provide the tool and training for creating libnfc-nxp.conf based on customer platform requirements. Please contact NXP support engineer for more details.

6. 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/` or `/system/vendor/lib/` on the android target system according to platform architecture. The firmware filename can be set in NXP_FW_NAME configuration in libnfc-nxp.conf file

Firmware can be updated when NXP releases a new 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

Note 1: Firmware download can take up to 20 seconds. Boot can take more time when FW is being downloaded.

Note 2: It is 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.

7. JCOP Download

NXP provides support for Jcop Os update feature for PN8xT samples over NFC wired mode (Only for PN80T) or SPI. nfcandroid_nxp_ese_clients folder (Corresponding library – ese_clients.so) contains required code to support this feature.

Steps to download Jcop OS are as follows:

1. Push the NXP released Jcop OS update files into folder /vendor/etc/JcopOs_Update1.apdu, /vendor/etc/JcopOs_Update2.apdu and /vendor/etc/JcopOs_Update3.apdu
2. Set the macro value NXP_ESE_JCOP_DEFAULT_INTERFACE = 0x01 for DWP and 0x02 for SPI interface in the libnfc-nxp.conf and push the updated config into /vendor/etc/
3. Reboot the device. New Jcop OS update will be downloaded during the NFC/eSE service boot up.

Note1: After successful JCOP OSU execution during boot, MW shall update jcop update state value 3 in metadata file (jcop_info.txt in /data/vendor/nfc or /data/vendor/secure_element based on interface selected).

Note2: Further reboots of the device shall not trigger the JCOP OSU as long as OSU state of MW metadata file is set to 3

8. Loader Service Update

8.1 Loader service in Boot Time

Loader service (LS) is used to install or update the applets on PN8xT over SPI. LS encrypted scripts resides at location /vendor/etc/loaderservice_updater_x.lss. (where x is enumerated from 1 to 10)

Please follow the below steps to integrate the loader service feature.

1. Push the encrypted applet files to /vendor/etc/loaderservice_updater_x.lss.
2. Reboot the device. New Loader service installs the applets during eSE service boot up.

Note1: Until the loader service scripts exist, ls update will be triggered on every reboot. LS scripts have to be compliant to multiple time execution and tearing safe.

8.2 Loader service in Run Time

The LS Device Agent (JAR) resides on the Rich OS layer of the device. It exposes an interface to Device Application's to execute LS script. The LS Device Agent is responsible for reading LS commands from LS script and forwarding them to the LS Application for processing and execution by the targeted Application. The LS Device Agent stores the response messages from the LS Application in a local file that can then be fetched by the Device Application

LS Agent is implemented as jar file(com.nxp.ls.jar). This jar is provided in

$NXP_GIT/nfcandroid_frameworks/sems/ and needs to be pushed to vendor/frameworks). Below are the 3 APIs provided as part of this jar. Note that Ls(Loader Service) is similar to Sems and so the API signature terms contain Sems.

- ***public static SemsAgent getInstance(Context applicationContext)***
  - **Description**: Returns LsAgent object. Creates new LsAgent if not created before otherwise returns existing LsAgent. **context** – Caller application context information
  - **Return value**: The LsAgent object
- ***int SemsExecuteScript(String inputscript, String filename, ISemsCallback callback) throws SemsException***
  - **Description**: Performs below following sequence.
    - Initializes Channel for APDU communication with eSE
    - performs secure script execution
    - return the status of execution of scripts via callback registered by LS Device Agent.
  - **InputScript** – Input LS script path provided by application
  - **FileName** - Name of the file in which output response APDU data to be stored
  - **semsCallback** - Callback provided by application which will be Invoked asynchronously once script execution is completed. This callback also provides the status of execution

- **Return value**:
- 0x00 - Success, Callback will be provided after execution is finished
- 0x01 – Failed – something is not correct, Script cannot be executed
- ***Throws****: LsException with string reason*
- ***Throws:*** *RuntimeException* If there are runtime exceptions

➢ ***String semsGetOutputData(String filename)***

- **Description**: Returns output response APDU of mentioned file name in string format to application
- Response output file will be deleted by LS HAL when new Script file is pushed. Application must ensure that output response data is retrieved before pushing next script file
- **fileName** – Response output file corresponding to given file name,
- **Return Value**: Output response data

Note1: The below change is required in the LS application with the MW version:11.02.0 onwards

a) In the application.java file import com.nxp.ls package.

9. Multi SE

Multi SE feature is used to switch AIDs from one Secure Element to another Secure Element which are present in the system. APIs as mentioned below are available to set/unset the route location for registered AIDs to available Secure Elements.

```
1    public boolean setOffHostForService(ComponentName service, String offHostSecureElement);
2    public boolean unsetOffHostForService(@NonNull ComponentName service);
```

More details on the API is available in link

10. Preferred Payment Service API

Preferred Payment Service feature is used to get AIDs and get route destination using the Preferred Payment Service. APIs as mentioned below are available to get route destination and registered AIDs to available..

```
Below 3 APIs are introduced to get more info about selected apk in Tap&Pay option
   a)  getAidsForPreferredPaymentService()
   b)  getRouteDestinationForPreferredPaymentService()
   c)  getDescriptionForPreferredPaymentService()
```

11. Enabling P2P

P2P is disabled by default from Android-11 builds. To enable it, “android.sofware.nfc.beam.xml” with contents below needs to be pushed to $ANDROID_ROOT/frameworks/native/data/etc in the build and /etc/permissions on the device.

```
<permissions>
    <feature name="android.sofware.nfc.beam" />
```

12. Support AID Power States Configuration

Nfc applications could use `requireDeviceUnlock` and `requireDeviceScreenOn` to configure the support power states for their AIDs.

1. `requireDeviceUnlock` or `requireDeviceScreenOn` attribute may be added in Payment app res xml file with value false or true. Default value be considered as false in case of attribute not set.

2. `requireDeviceScreenOn` defines whether AID’s registered part of this service can work in Screen off state or not, flag set to false allows payment app to work in screen off state.

13. Field Detect Feature

Filed detect feature is used to disable the Passive Listen Phase and block the RF activation for listen mode from the application layer. When the field detect feature is enabled passive listen mode phase is disabled and proprietary command (FF01) sent in the discovery command to receive the RF field detect notification. The RF activation will be blocked until the field detect feature is disabled. The field detect feature settings are not persistent and will be overwritten with NFC OFF/ON or NFC service kill or device reboot. APIs as mentioned below are available to enable/disable field detect mode and get the field detect mode status..

```
1    public int setFieldDetectMode(boolean mode)
   -   mode: enable if true , disable if false
2    public boolean isFieldDetectEnabled()
```

14. FeliCa Middleware Conformance Test Configuration

FeliCa Middleware Conformance Test (FMCT) specification is defined by FeliCa Networks for handset. The below settings are necessary for compliance with FeliCa requirement specifications.

FMCT is applicable only for PN81x chipset.

14.1 FMCT Configurations in libnfc-nci.conf files

Configurations	Value	Descriptions
P2P_LISTEN_TECH_MASK	Ox41	Disable to use NFC-F for Android Beam

14.2 FMCT Configurations in libnfc-nxp.conf file

Configurations	Value	Descriptions
NXP_CORE_CONF_EXTN	A0 80 02 FA 00	Set timeout to keep listen after RF off to 250ms
NXP_CORE_CONF_EXTN	A0 98 01 07	Enable fix for CLT issue
DEFAULT_SYS_CODE_PWR_STATE	0x3B	Enable the SCBR power state

15. T4T NDEF Emulation from eSE.

T4T NDEF java card application emulates the T4T NDEF TAG operations. This application shall be installed inside embedded Secure Elements (eSE) integrated in a Mobile Device. Steps to update the NDEF data to be emulated is as below:

1) Compose NDEF message as per NDEF protocol specification
2) Using OMAPI API from android select the required applet and write NDEF message using commands specified by applet
3) Corresponding CLT parameters need to be enabled by using commands specified by applet.
4) Set the “DEFAULT_AID_ROUTE” to eSE in libnfc-nxp.conf file
5) Reboot the emulator device
6) Tap Reader device and select T4T card application and read NDEF content

Note: Pre-requisite T4T applet should be installed using specification provided by applet

16. GSMA Conformance Test Configuration

GMSA Handset Test Book Specification is defined by GSMA and Global Platform defines the OMPAI specifications. GCF / PTCRB defines the test cases based on the GSMA & GP OMAPI. Settings mentioned below are necessary for compliance with GSMA Handset Test Book Specifications.

16.1 GSMA Configurations in libnfc-nci.conf files

Configurations	Value	Descriptions
P2P_LISTEN_TECH_MASK	0x84	SAK value is expected 20. To be enabled for GSMA test case ID: 7.3.8.11
NXP_PROP_BLACKLIST_ROUTING(PN80T)	0x01	To be enabled for GSMA test case: 15.7.3.10.1

16.2 GSMA Configurations in libnfc-nxp.conf file

Configurations	Value	Descriptions
NFC_DEBUG_ENABLED	0x00	Log contains data (AID: AID01) generated by the card.
NXP_PROP_BLACKLIST_ROUTING(PN80T)	0x01	To be enabled for GSMA test case: 15.7.3.10.1

17. DTA APK User Manual

17.1 Introduction

Device Test Application (DTA) that a vendor can integrate in an NFC Forum Device to ensure that the Implementation/Device Under Test (IUT/DUT) can be tested against the NFC Digital Protocol Technical Specification [DIGITAL], NFC Forum Type 1-4 Tag Operation Specifications [T*n*TOP], NFC Forum Analog RF, LLCP and SNEP.

DTA APK is designed to work with NCI based NFC chipsets. This setup guide provides the detailed directions about setting up NFC DTA apk for NFC Forum Compliance Testing of Implementation Under Testing (IUT) or Device Under Testing (DUT).

17.2 Scope

This document is written considering NFC DTA apk setup guidelines to perform the NFC Forum compliance validation of Implementation Under Testing (IUT) or Device Under Testing (DUT).

17.3 Architecture of NFC DTA APK

Figure 3 shows the architecture of NFC DTA APK.

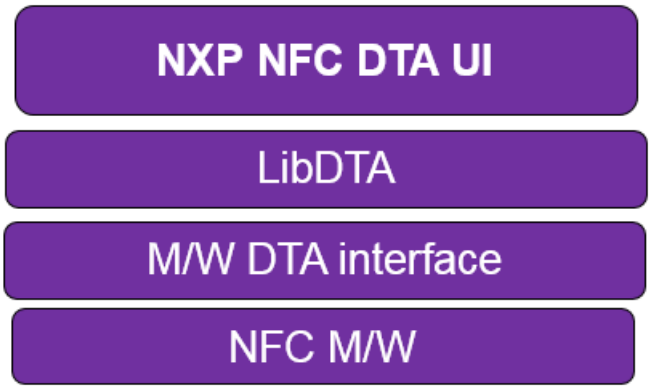


Figure 3: NFC DTA APK Architecture

17.3.1 NFC DTA supported Features:

A NFC device may support one or more communication technologies: Type A, B and F, in both Poll & Listen modes.

17.3.2 Testing Scope

- NFC Forum Digital protocol test cases.
- NFC Forum T1T, T2T, T3T & T4T test cases
- NFC Forum Analog RF.
- NFC Forum LLCP.
- NFC Forum SNEP.

17.4 NFC DTA Setup

17.4.1 NFC DTA Source

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

Repository Name	Checkout Command
NXPAndroidDTA	git clone https://github.com/NXPNFCProject/NXPAndroidDTA.git The contents of this folder needs to be placed in packages /system/nfc-dta/ directory in the android build.

17.4.2 Build NFC DTA

Copy the nfc-dta souce files to directory AROOT/system/nfc-dta. Build the DTA.

After compilation it generates 64bit 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.

17.4.3 NFC DTA Source

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.

Settings to be done before running DTA APK are as below

- Switch off the default NFC service option in Settings.

  Settings->Connected Devices >NFC as OFF (Un-ticked) and reboot the device (using ‘adb reboot’).

- 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.

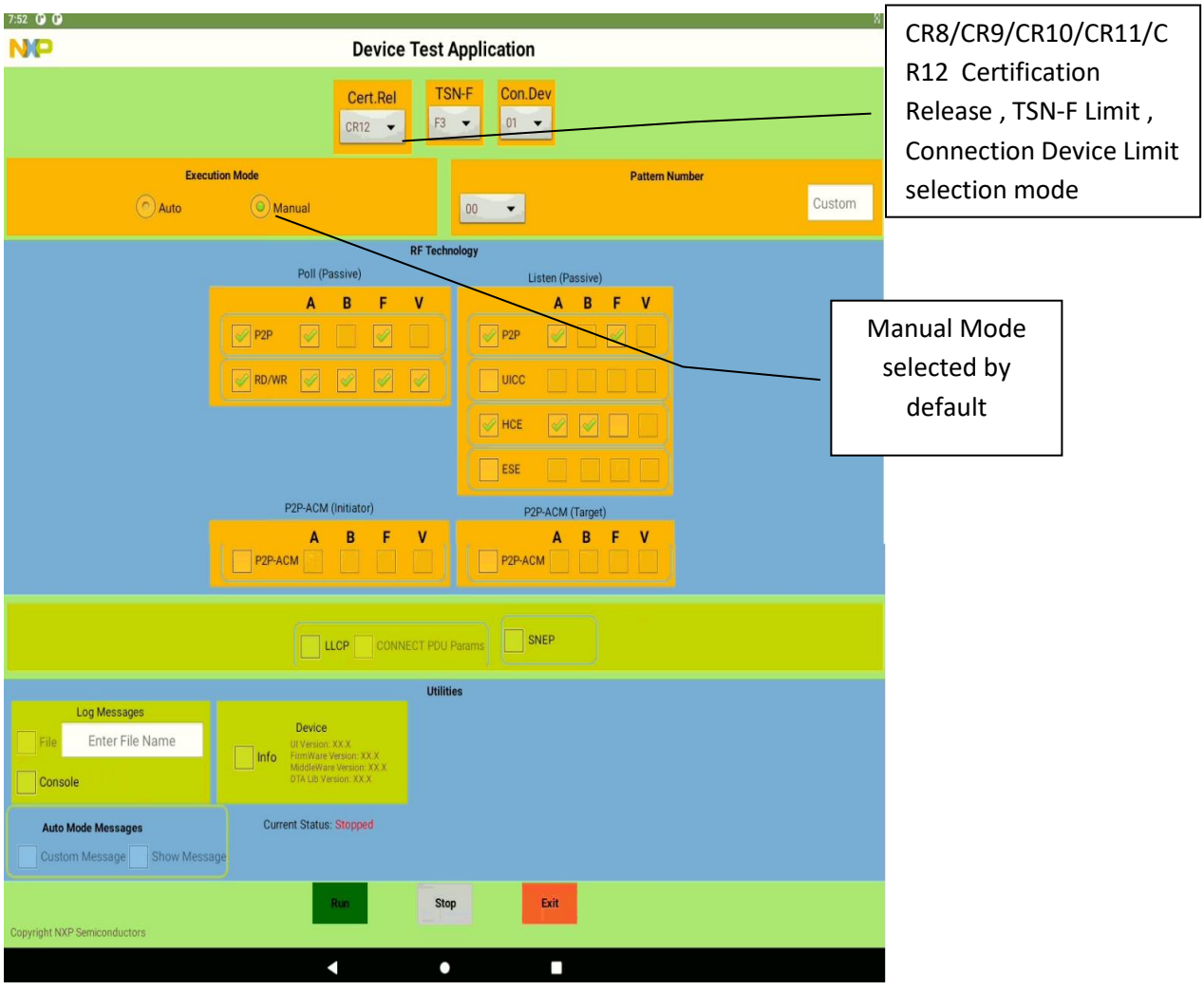
Settings -> Display -> Sleep -> select 30 minutes.

17.5 DTA APK Menu selection

17.5.1 NXP_DTA_UI_SCR_SCENERIO_01: Default screen

The default screen is loaded as soon as the application is launched. By default certification release CR12 with Time Slot Number(TSN-F) for NFC-F technology with value F3, Connection Device Limit(Conn.Dev) with value 01 and manual mode is selected and the pattern number will be set to “00” in multi option. The user has the option to enter custom pattern number.

By default some of the RF Technologies will enabled for both Poll & Listen modes. Device information will not be displayed in the default screen. The current status of the application is “stopped” and the text color is in red. The RUN button in GREEN color, STOP button in GRAY color and EXIT button in orange color. In manual mode check boxes Custom Message and Show Message are disabled. Copyright and UI Version are showed in the bottom. “Manual” Mode is selected by default. “Auto” mode is added for future extensions. Before running the DTA application select Connection Device Limit Con.Dev to value 01.

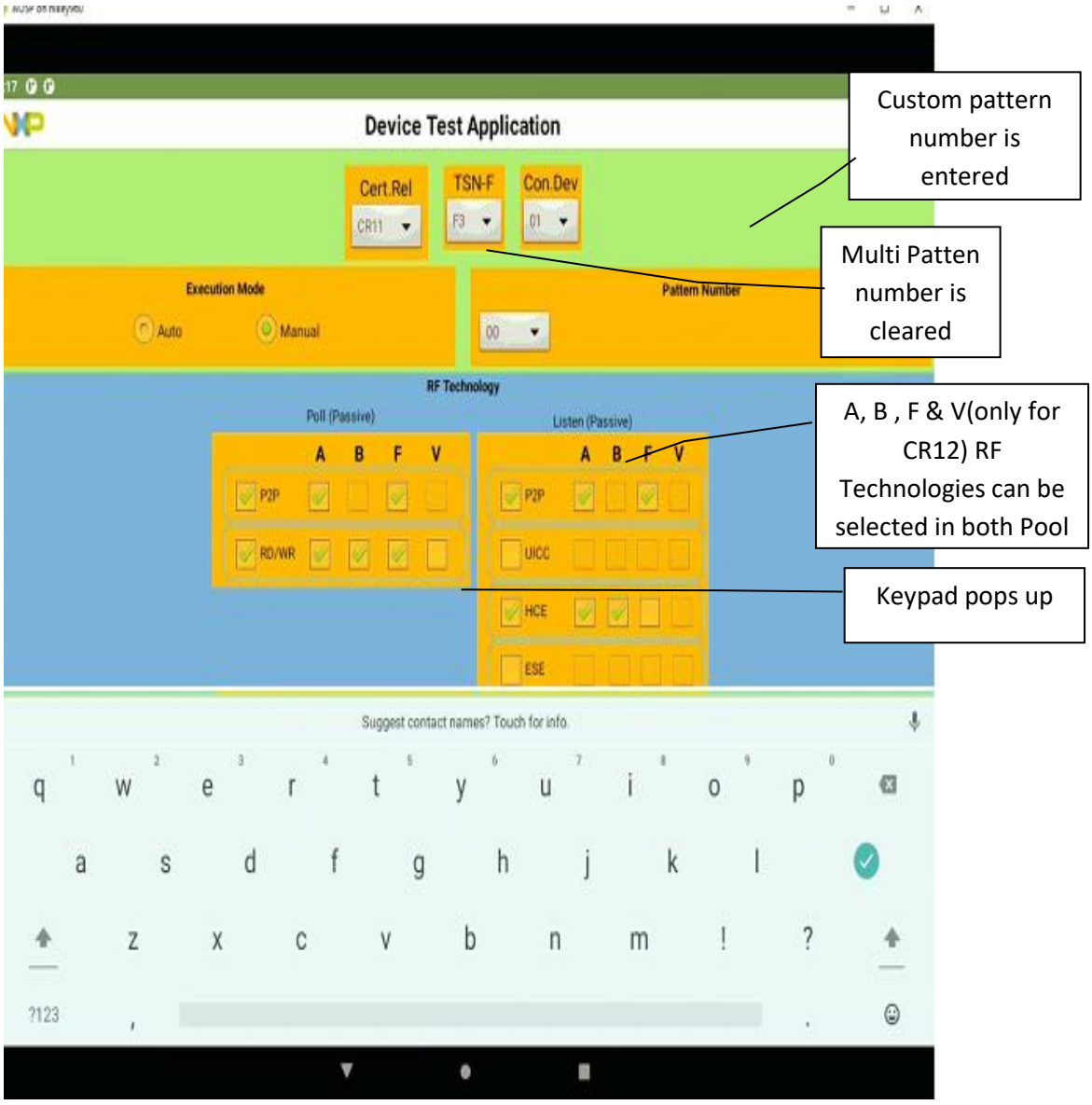


17.5.2 NXP_DTA_UI_SCR_SCENERIO_02: Selections in Manual Mode

This screen is similar to “NXP_DTA_UI_SCR_SCENERIO_01” screen with the changes shown based on the user selection.

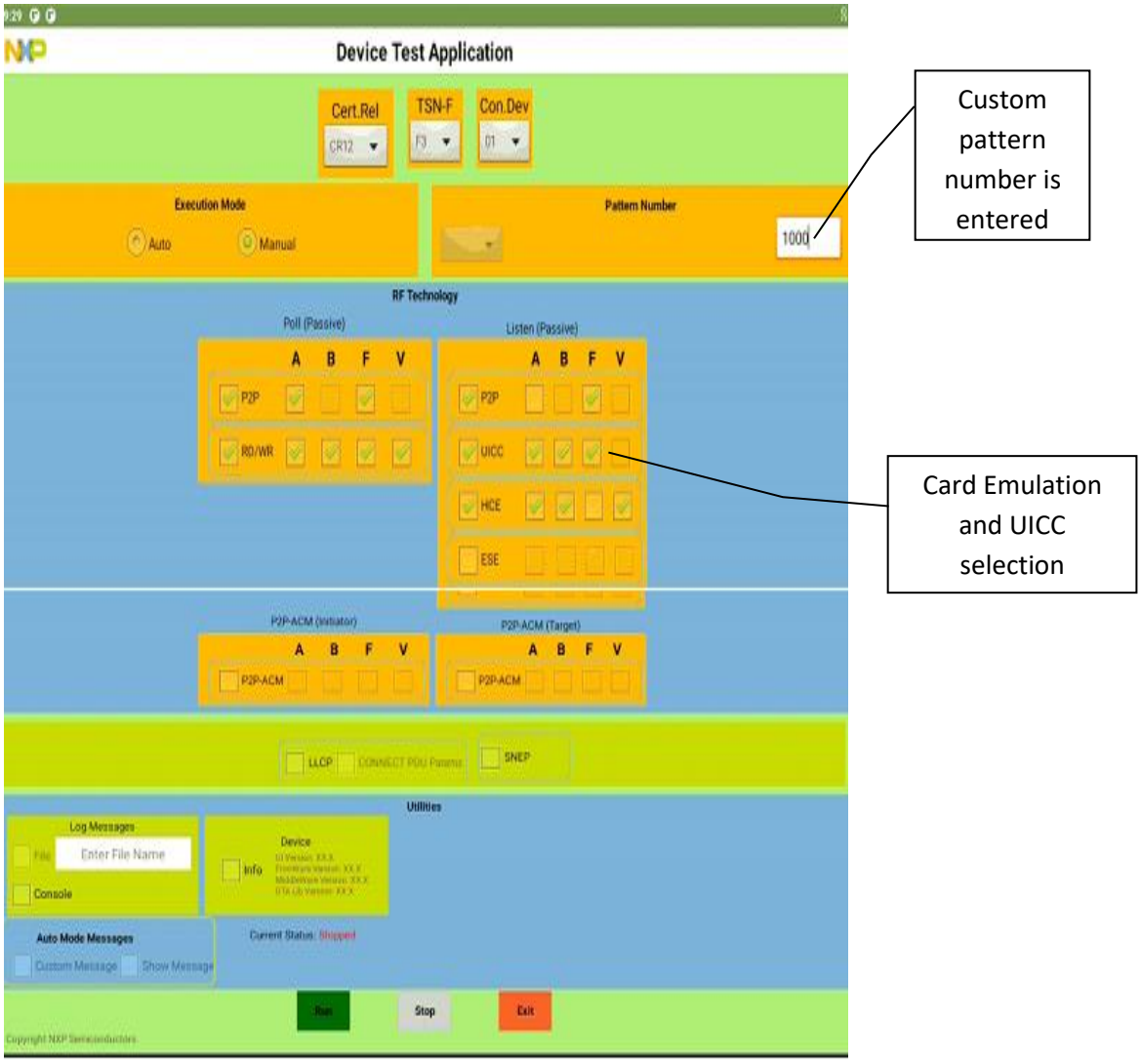
The custom pattern number is entered as 0000. Multi pattern number if selected will get cleared. Need to enter hexadecimal pattern number without the prefix 0x. Only 0000 to ffff is allowed to enter. Other entry will show pop-up message as shown in the. Maximum number of bytes allowed is only 4. As soon as the user touches in the custom pattern number box, the keypad pops up as shown in the screen.

Below are RF Technology options available for selection in Poll & Listen mode. In Poll Mode P2P and Rd/Wr modes are allowed to select. However enabling one technology in one of the poll modes will enable the same technology in other poll mode. Listen mode P2P, UICC, HCE and ESE are allowed to select. In LLCP, parameters in CONNECT PDU is allowed to be enabled/disabled.



17.5.3 NXP_DTA_UI_SCR_SCENERIO_03: Analog Selection in Manual Mode

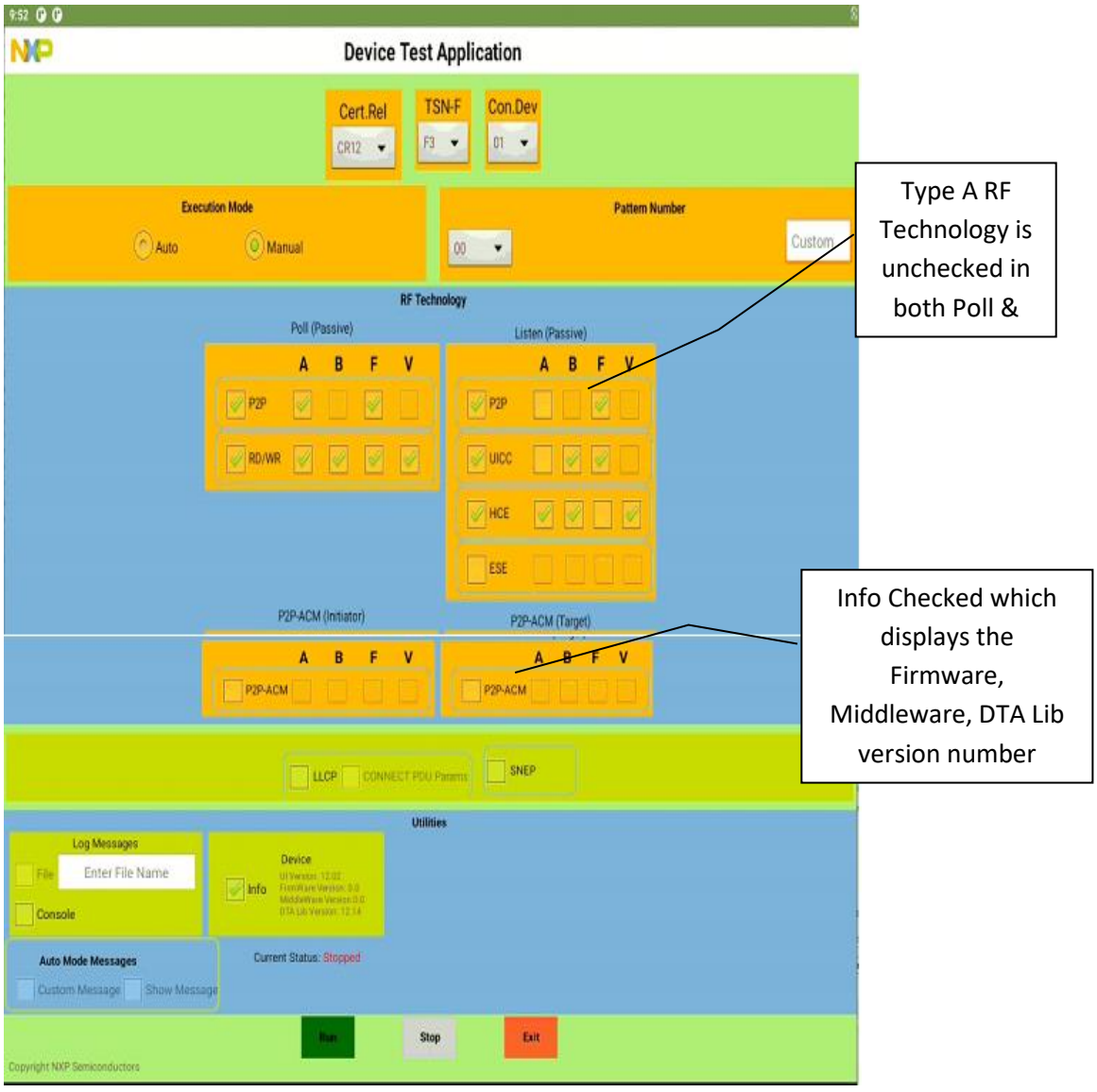
First select the CE mode with UICC. The custom pattern number is entered as 1000 and press RUN button then the application will start running in manual mode. The current status will be changed to first Running and the text color is in green. Now, all the selections are disabled.



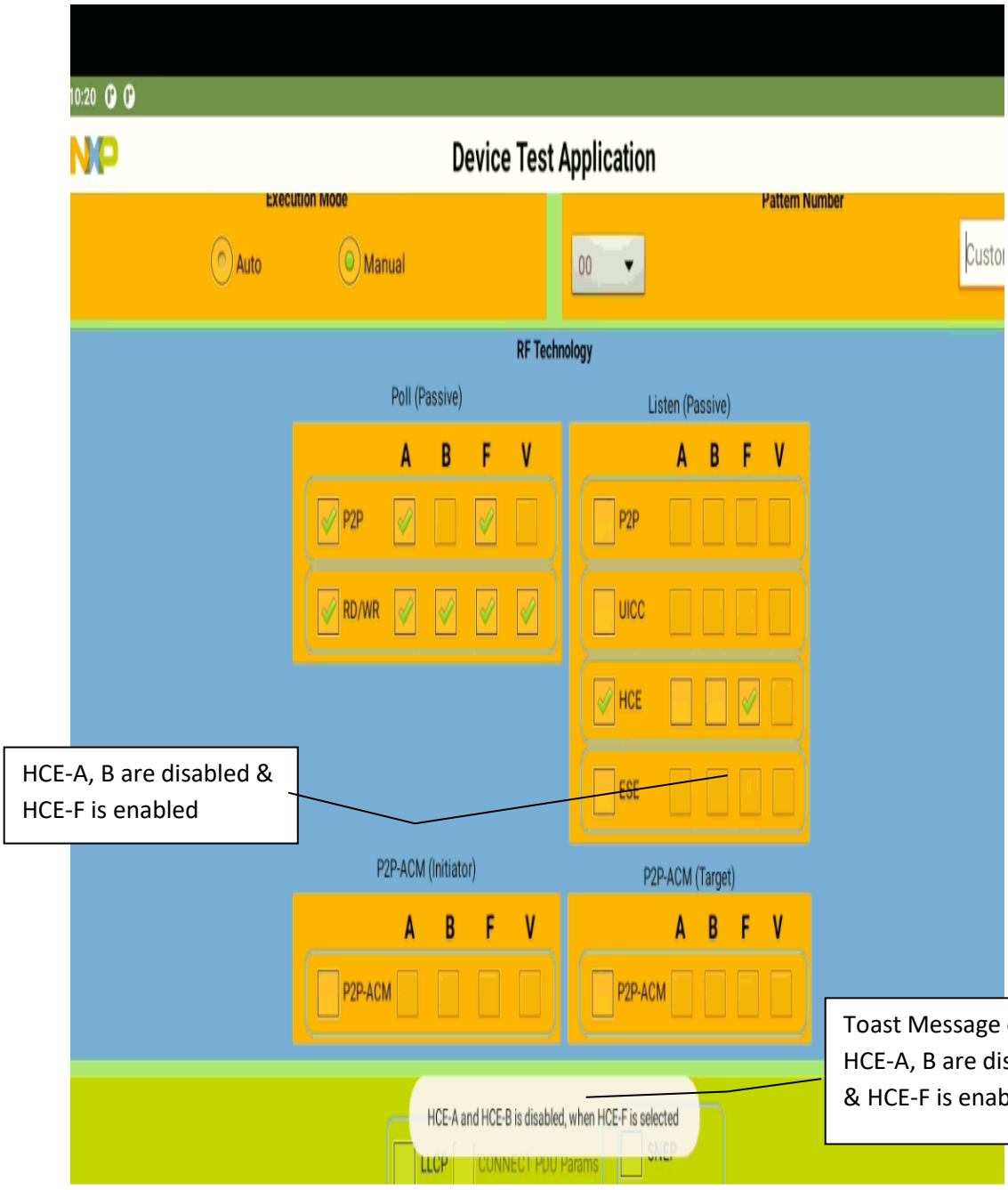
17.5.4 NXP_DTA_UI_SCR_SCENERIO_04: De-selections in Manual Mode

The application Current Status should be Stopped for de-selections. In this screen the user has selected unchecked the check box B in the RF Technology both in Poll and Listen mode. If all the technologies in a test mode is unchecked then the corresponding test mode is unchecked.

The Device Info check box is checked, which shows the device information as soon as it is checked. If the user unchecks, then the info will not be shown.



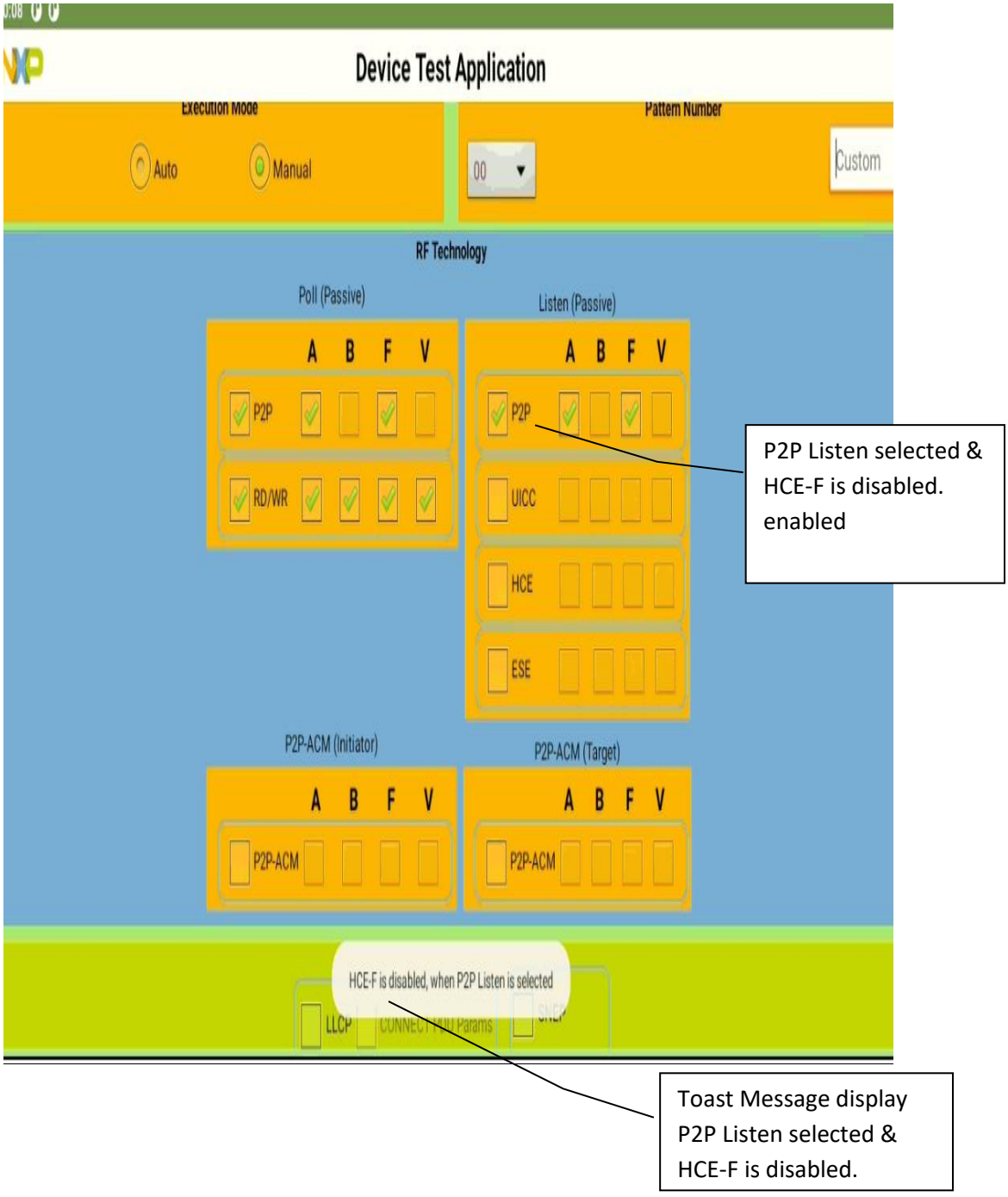
17.5.5 NXP_DTA_UI_SCR_SCENERIO_05: UI toast messages for HCE NFC-F:



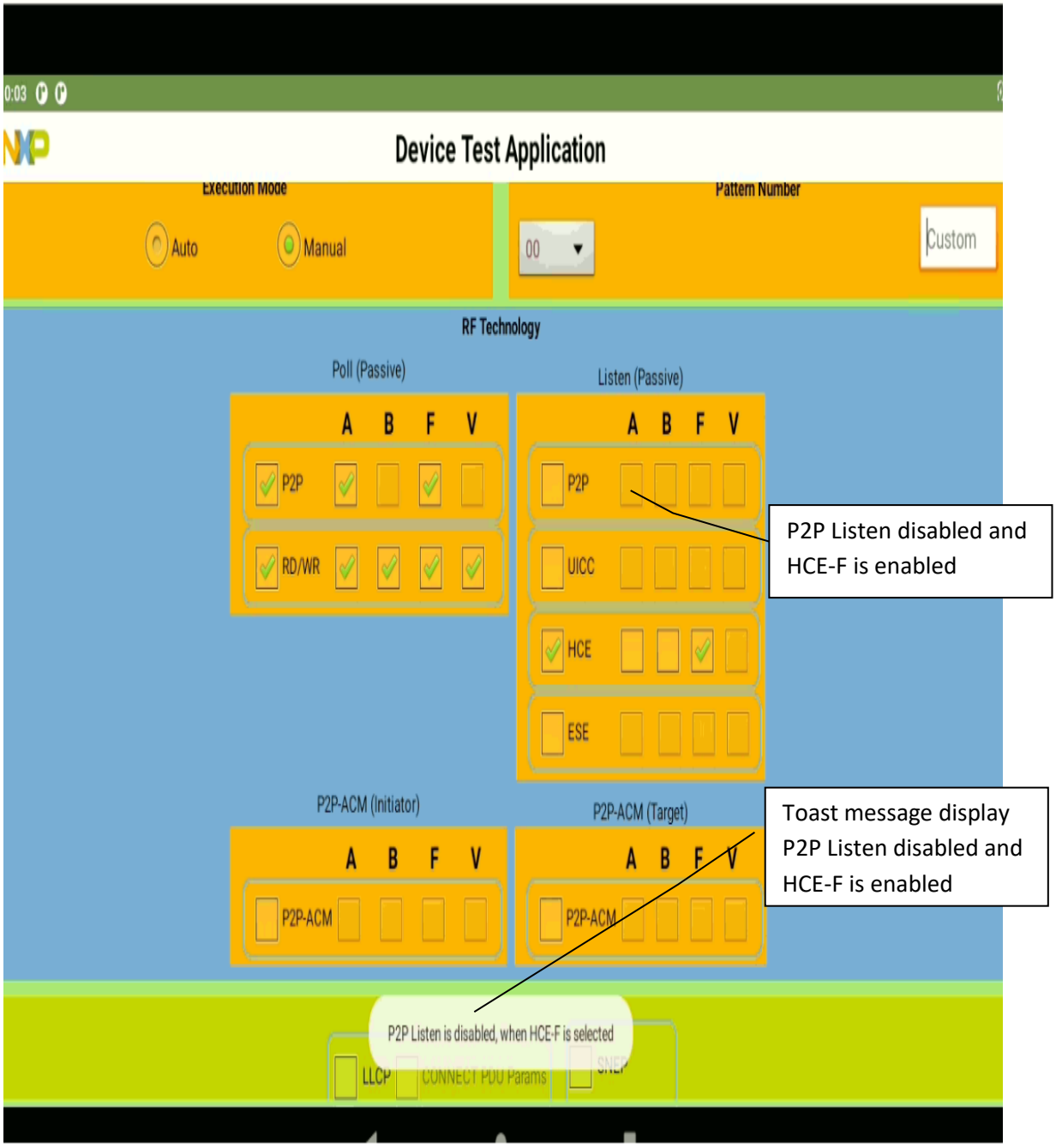
To run the Listen mode HCE NFC-F test case need to disable HCE NFC A & NFC B technology & enable HCE NFC F technology in Listen Mode.

Toast message display P2P Listen disabled and HCE-F is enabled

To run P2P test cases in Listen mode, deselect HCE NFC-F.



When HCE NFC-F technology enabled in DTA application, it will disable P2P A & F technology with Toast message.

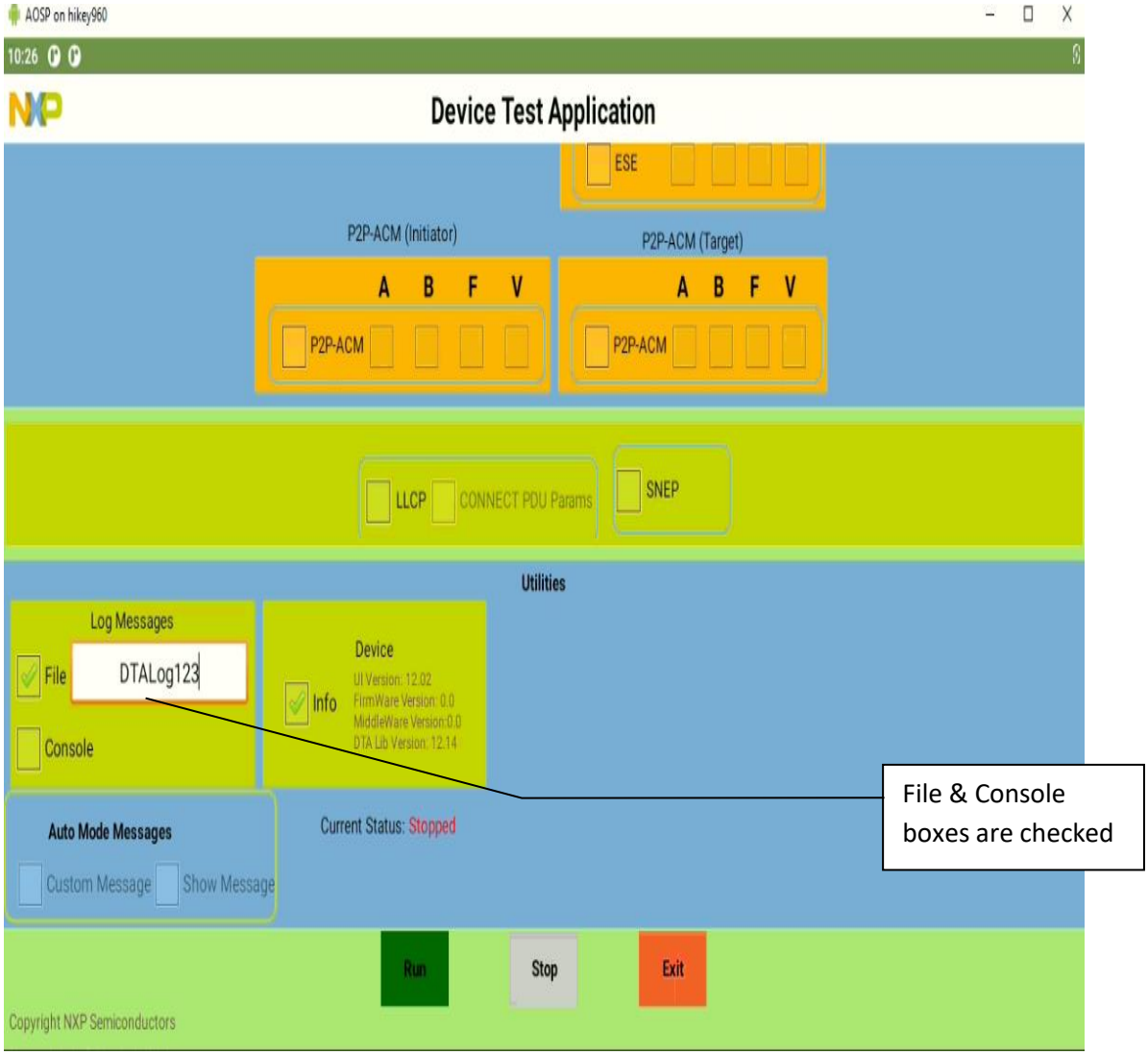


17.5.6 NXP_DTA_UI_SCR_SCENERIO_05: Device Info and Log Messages

Similar to the screen “NXP_DTA_UI_SCR_SCENERIO_02”, but the user is selecting the device type from the drop down list. The device selection is ranging from 00 – 05. Also the user has checked the Info box, which shows the device version info.

The user has deselected the RF Technology type B in both Poll & Listen modes.

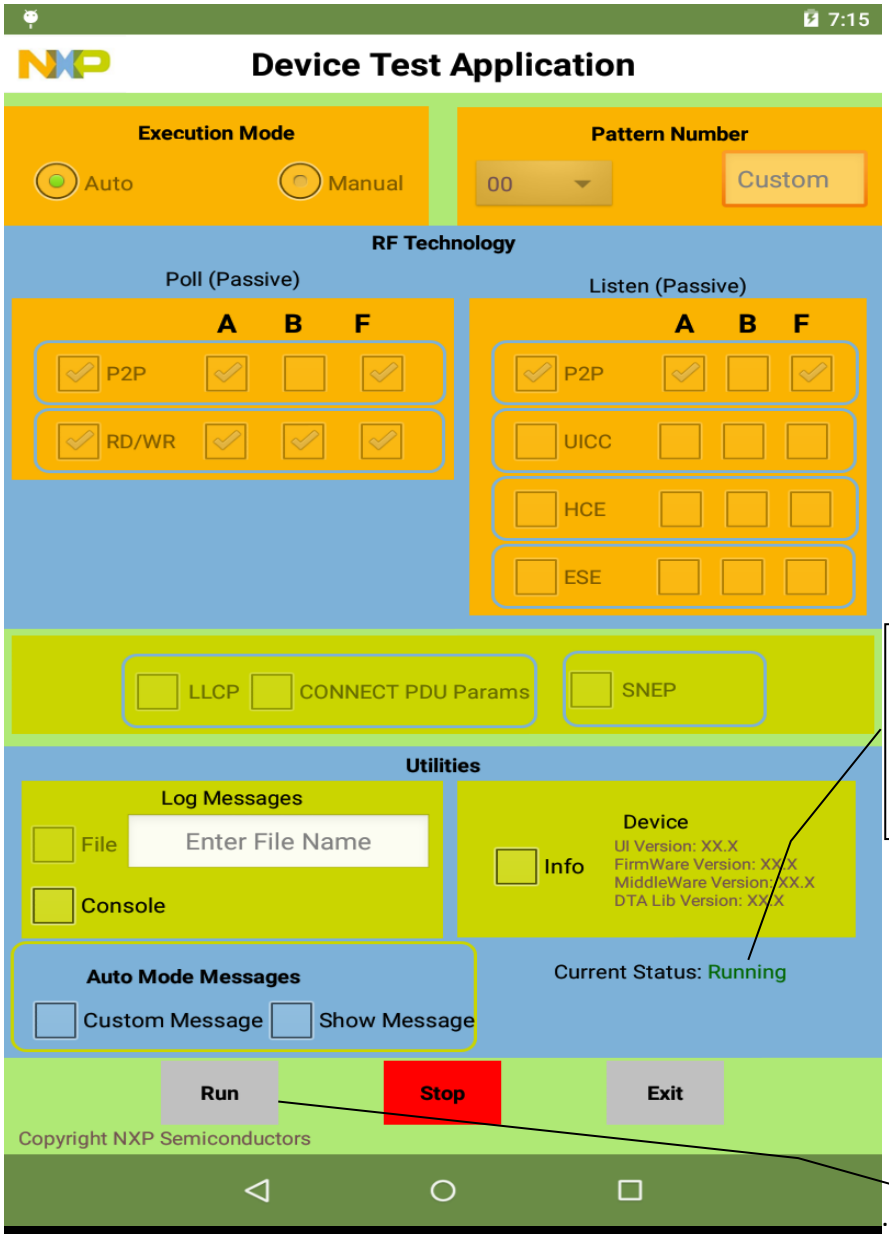
Also the user has checked the console box to see the system message logs on a separate console. The user has entered the file name and checked the File check box to write system log messages on to the SD card. By default the system logs will be stored in the directory “/sdcard/nxpdtalog/”. User need not to enter the file extension. By default it will be stored as “.txt” In the below case it logs will be stored in “sdcard/nxpdtalog/dtaLOG123.txt”. If the user clear the file name by pressing back button then the check box will be unchecked automatically.



17.5.7 NXP_DTA_UI_SCR_SCENERIO_07: Running in Manual Mode

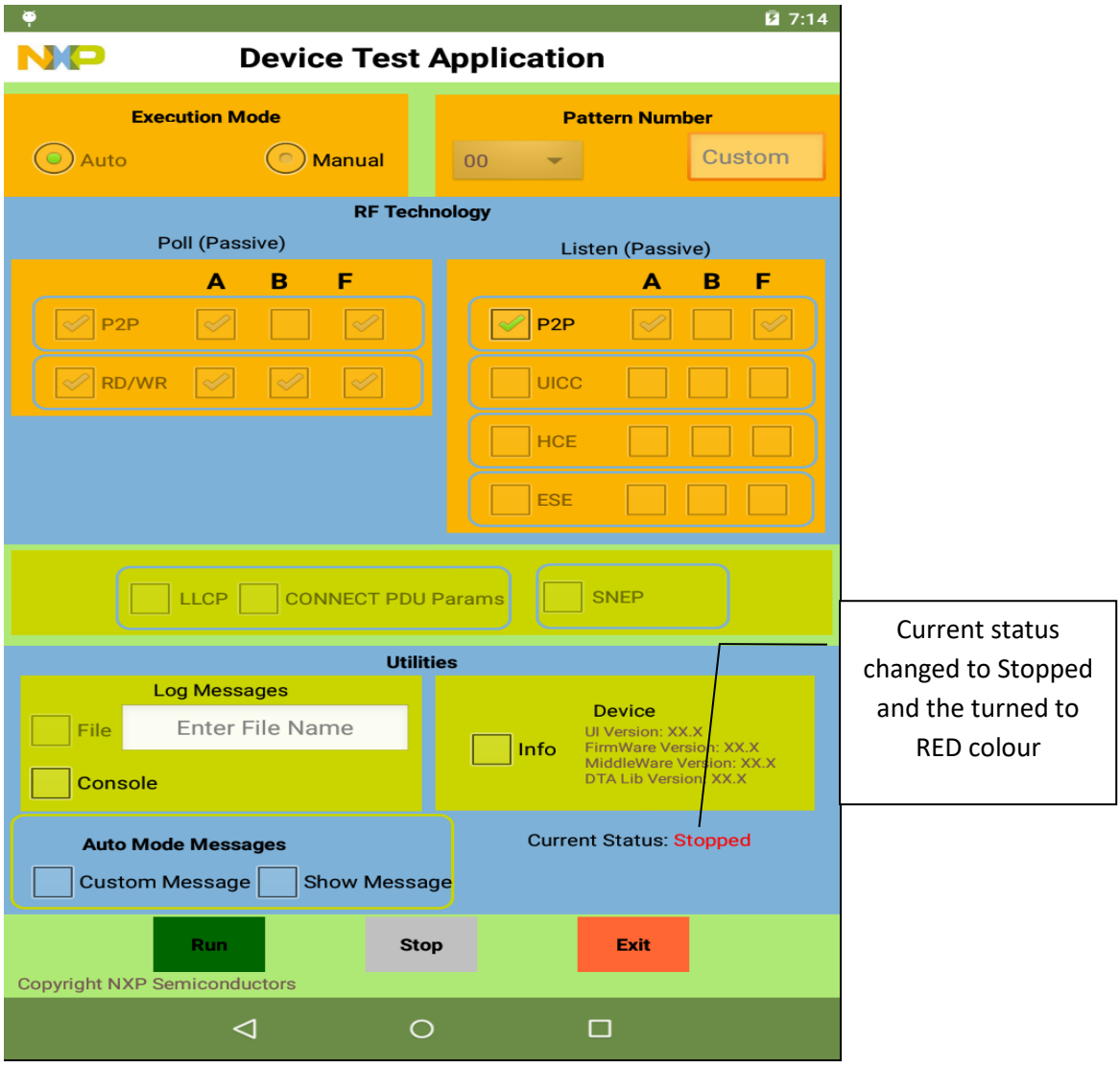
IN this screen, the user press RUN button then the application will start running in manual mode. The current status will be changed to Running and the text color is in green. During running, no other selections are allowed.

The RUN button will turn to GREY color. STOP to RED color & Enabled to use. EXIT to GREY color.



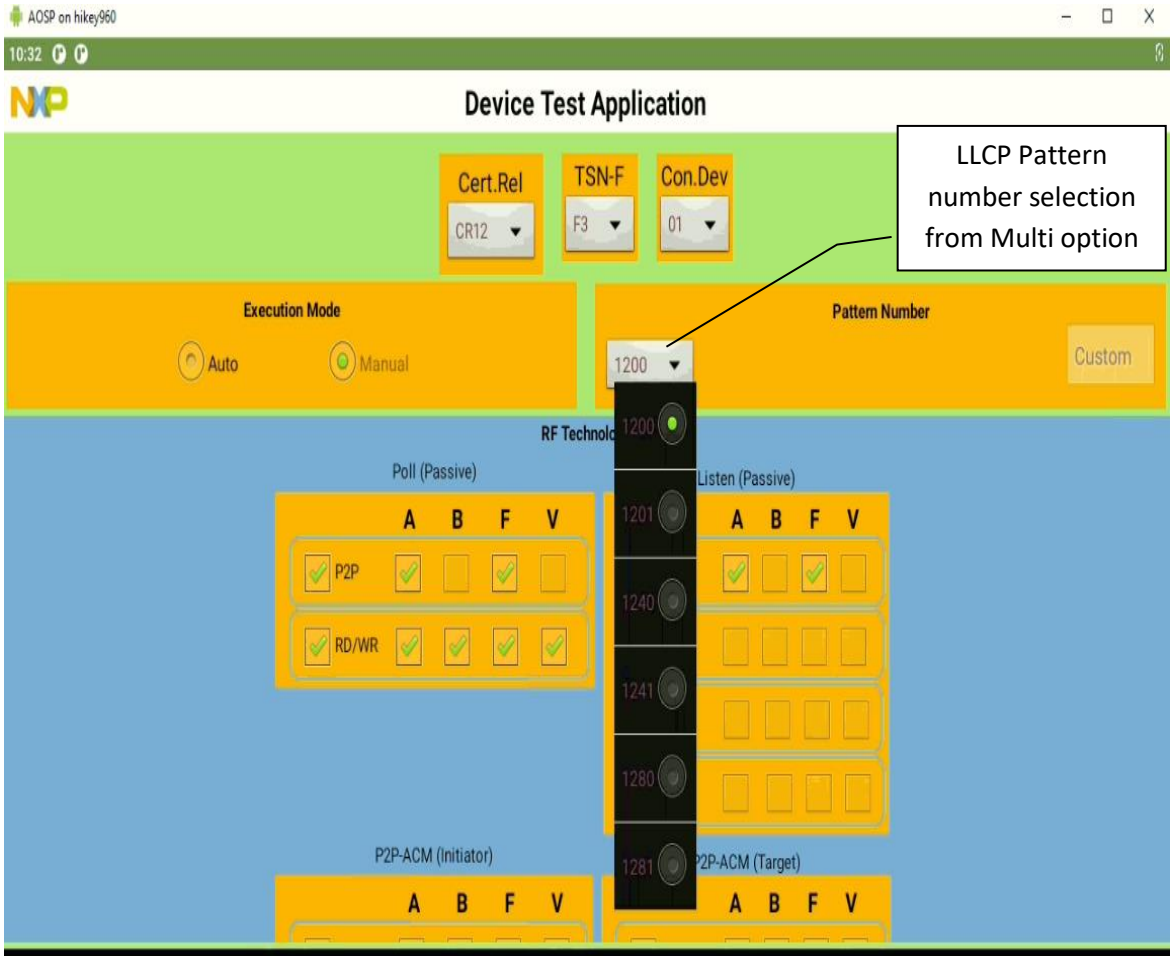
17.5.8 NXP_DTA_UI_SCR_SCENERIO_08: Stopping in Manual Mode

In the previous screen the user pressed Stop button to stop the application. Now the application has stopped as shown in the current screen. The current status is changed to Stopped and is in red color. Now all the selections are cleared (and will remain in manual mode) and enabled for selection except the Custom Message and Show Message check boxes.



17.5.9 NXP_DTA_UI_SCR_SCENERIO_09: LLCP Setting.

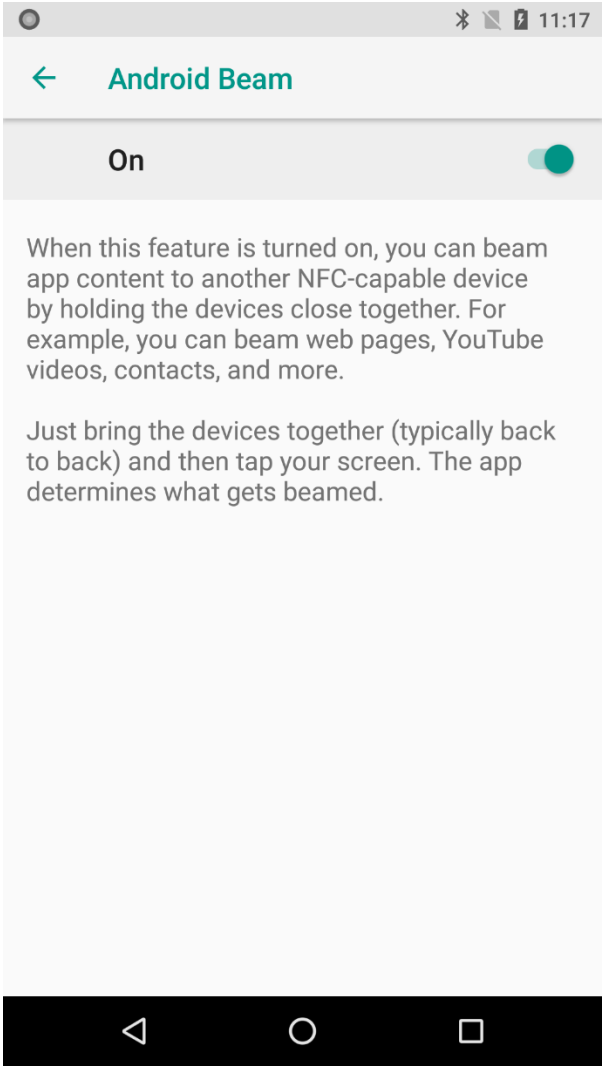
Select LLCP from DTA to run the LLCP test cases. LLCP pattern numbers are enabled in Multi pattern drop down list. Select the appropriate pattern number and press RUN button then the application will start running in manual mode. The current status will be changed from stopped to Running and the text color is in green. Now, all the selections are disabled. Parameters in CONNECT PDU are enabled by default when LLCP is selected. Disable this option if IUT is not required to send parameters in CONNECT PDU.

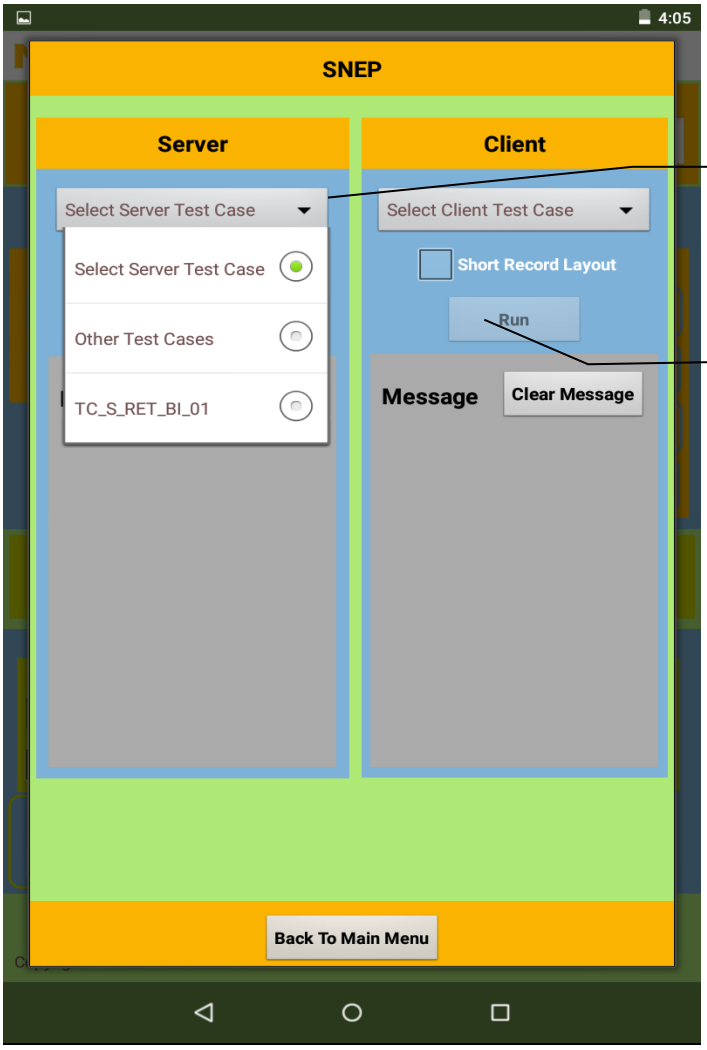


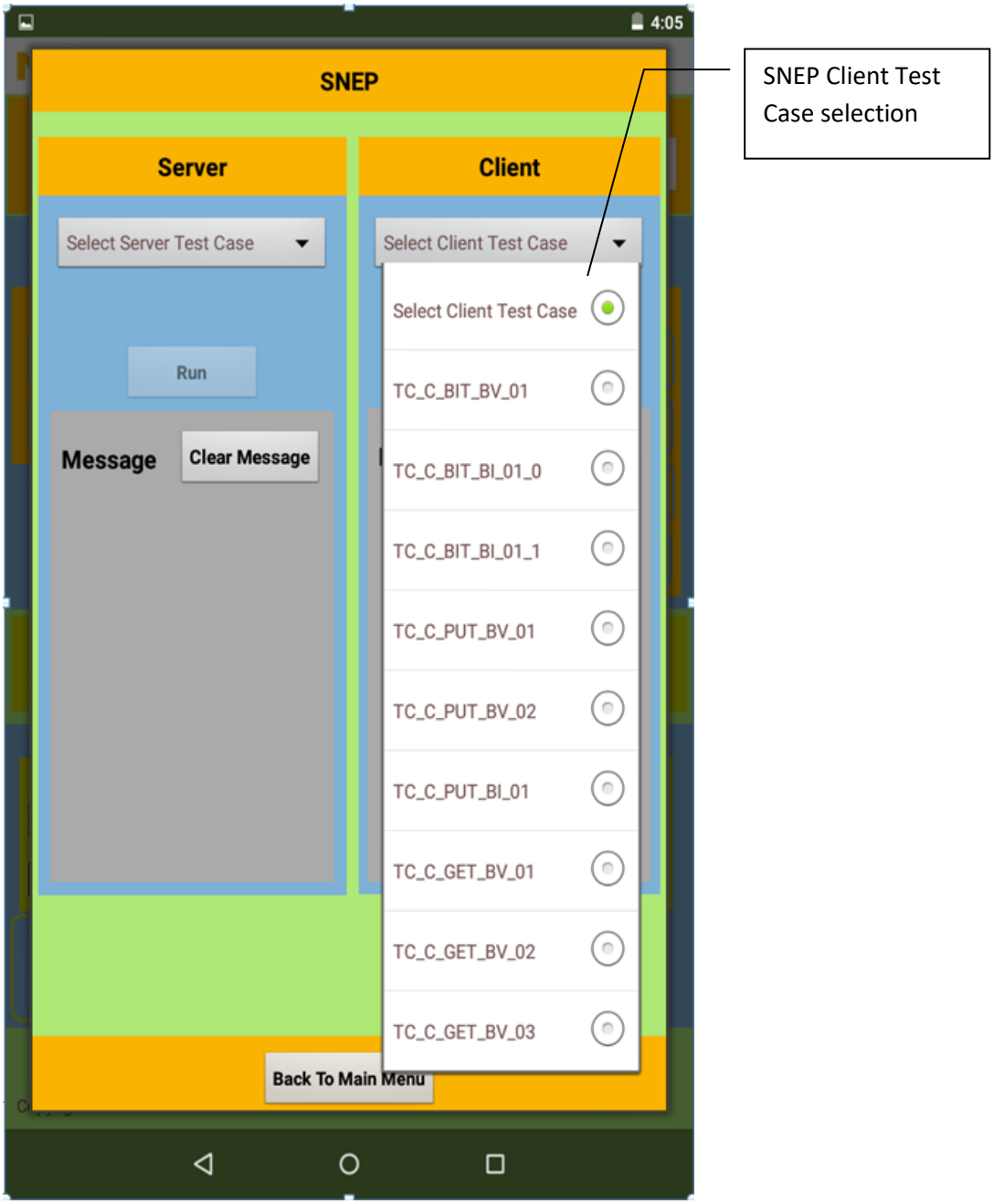
17.5.10 NXP_DTA_UI_SCR_SCENERIO_10: SNEP

To Run the SNEP test cases, please follow the steps below

- Select SNEP to run the SNEP test cases. As soon as select the SNEP button, SNEP setting screen will open. Select SNEP Server test case and Press the Run button to start SNEP server.
- Android beam shall be set to ON: Settings->Connected Devices -> Android Beam







- Select the SNEP client test case ID and press RUN.
- The selected SNEP client test case ID settings will applicable only for particular selected test case ID only not for other test case ID's.
- Press **Back to Main Menu** button to come back from SNEP screen to Device Test Application.

18. Legal information

18.1 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.

18.2 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.

18.3 Licenses

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

18.4 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>

18.5 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.

Contents