AN11762

Android NFC Setup Guide

Rev. 1.5 — 13 September 2017 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.

Contact information

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

1. Introduction

NXP's NCI based NFC controllers (PN547C2/PN548C2/PN553/PN65T/PN66T/PN80T/NQ210/NQ220/NQ310/NQ330) are designed to work with Android open source using NCI based stack for Android NFC. In the further sections, NXP-NFCC refers to PN547C2/PN548C2/PN553/PN65T/PN66T/PN80T/NQ210/NQ220/NQ310/NQ330.

2. Scope

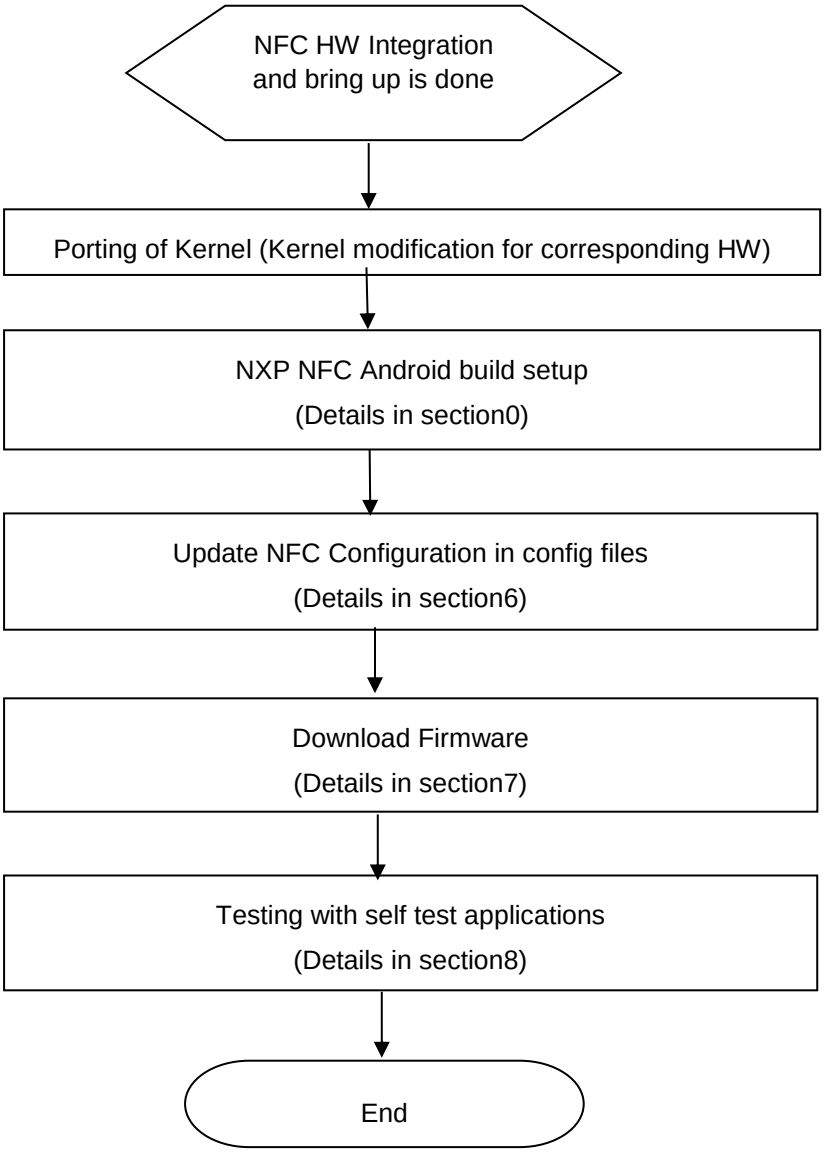
This setup guide provides guidelines for setting up NXP NFC in android build environment. It is an example guideline for basic system integration. OEM integration may have variations based on actual system and integration.

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.



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.

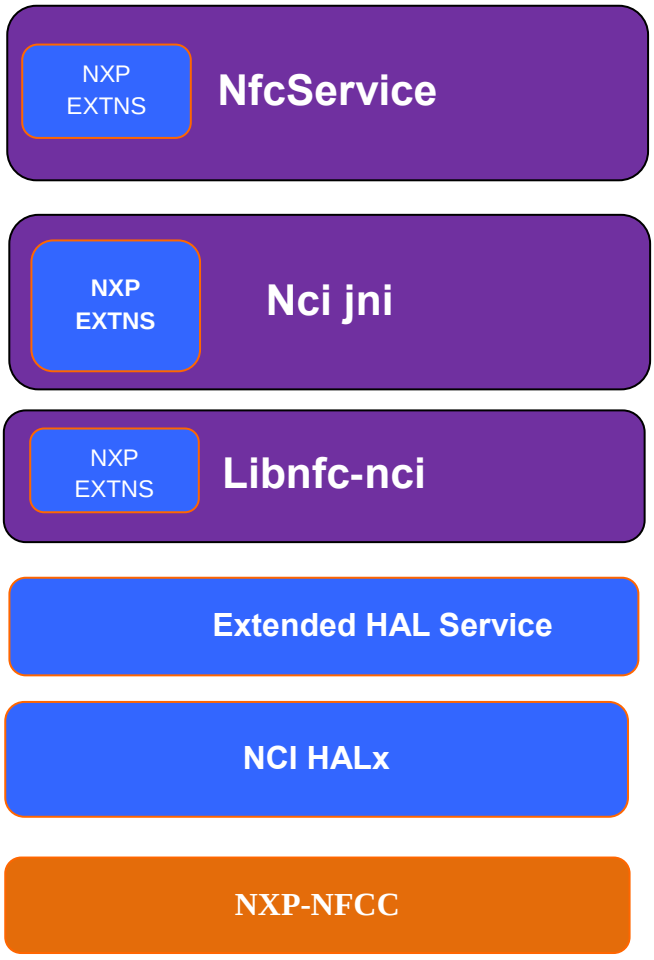


Figure 2: Android NFC with NXP-NFCC

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 external/libnfc-nci directory in the android build.
NFCNCIHAL_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.
NXPNFCC_FW	*git clone https://github.com/NXPNFCProject/NXPNFCC_FW.git*
NXPAndroidDTA	*git clone* https://github.com/NXPNFCProject/NXPAndroidDTA.git More details about installation of DTA is in section 9.4 9.4

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	Contains Android NFC Framework files
	NFC_NCIHAL_base/core/java /com/nxp/nfc NFCNCIHAL_base/core/java/com/vzw/nfc	NXP's extensions to android NFC Framework implemented for Android N and earlier versions
	NXPNFC_Reference/vendor/nxp/opensource/frameworks	NXP's extensions to android NFC Framework implemented only for Android Oreo
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	NFC_NCIHAL_libnfc-nci/halimpl/pn54x	Contains hardware abstraction layer for NXP specific controllers
	NXPNFC_Reference/vendor/nxp/opensource/hardware/interfaces	Contains HAL service layer for NXP specific controllers only for Android Oreo
Firmware	NXPNFCC_FW/PN54X/32-Bit/libpn5xx_fw.so NXPNFCC_FW/PN54X/64-Bit/libpn5xx_fw.so	It is a directory, which includes the firmware file for NXP-NFCC

5.4 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
NXPNFC_Reference/vendor/nxp/opensource/frameworks (only for Android Oreo)	Will be part of com.nxp.nfc.jar	/system/framework
NFC_NCIHAL_Nfc/nci	libnfc_nci_jni.so	/system/lib
	NfcNci.apk	/system/app/NfcNci
NFC_NCIHAL_libnfc-nci	libnfc_nci.so	/system/lib
	nfc_nci_pn54x.default.so	/system/lib/hw
NXPNFC_Reference/vendor/nxp/opensource/hardware/interfaces (only for Android Oreo)	vendor.nxp.nxpnfc@1.0-impl.so	/system/vendor/lib/hw
	vendor.nxp.nxpnfc@1.0-service	/system/vendor/bin/hw

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

5.5 Building the Kernel Source Code

Follow the following steps for building the kernel:

- Create nxp/pn6xT and nxp/pn8xT folder inside kernel/driver/
- Copy pn54x-i2c from NXPNFC_I2CDriver and keep inside as per chip type.
- Copy p61-spi and p73-spi from NXPNFC_SPIDriver put inside the nxp/pn6xT and nxp/pn8xT with respect to the chip type.
- 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

6. Compilation of NCI HALx and Extensions

6.1 Compilation Flags

Below are some of the compilation flags defined in Android.mk files. Note that these are applicable for Android N. For Android Oreo, support for common MW with single binary does not follow these macros as the chip information is captured during boot time.

Compilation Flags	Description
NFC_NXP_CHIP_TYPE	Choose the appropriate NFC chip used in the system. By default PN553 is enabled as below #variables for NFC_NXP_CHIP_TYPE PN547C2 := 1 PN548C2 := 2 PN551 := 3 PN553 := 4 NQ110 := $PN547C2 NQ210 := $PN548C2 NQ310 := $PN553 (1)external/libnfc-nci/Android.mk (2)external/libnfc-nci/halimpl/pn54x/Android.mk (3)packages/apps/nfc/nci/jni/Android.mk D_CFLAGS += -DNFC_NXP_CHIP_TYPE=PN553 D_CFLAGS += -DNFC_NXP_CHIP_TYPE=PN551 D_CFLAGS += -DNFC_NXP_CHIP_TYPE=PN548C2 D_CFLAGS += -DNFC_NXP_CHIP_TYPE=PN547C2 **For Android Orea with latest MW tag chipe type macros are not used,hanled with run time.**

NFC_NXP_ESE	Set to false in Android make files as referred in the NFC_NXP_CHIP_TYPE compilation flag if Embedded secure element is not used.

6.2 Configuration files

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

6.2.1 Configurations in `libnfc-brcm.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

6.2.2 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_DESFIRE_ROUTE	Configuration to set default route location for ISO-DEP Protocol. This settings will be used when application does not set this parameter using the MifareDesfireRouteSet() API defined in android framework. • Host: 0x00 • eSE: 0x01 • UICC: 0x02 • Default: 0x02
DEFAULT_MIFARE_CLT_ROUTE	Configuration to set default route location for A, B & F Technology. This settings will be used when application does not set this parameter using the MifareCLTRouteSet() API defined in android framework. • Host: 0x00 • eSE: 0x01 • UICC: 0x02 • Default: 0x02
NXP_FW_NAME	Name of the firmware file (ex: libpn5xx_fw.so) . This name shall be name of the file as in /system/vendor/firmware directory

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.

7. 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/firmware/ 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 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/firmware 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.

8. Self-test API description

This section describes the various self-test API's provided by NXP.

API	NFCSTATUS phNxpNciHal_TestMode_open (void)
Description	It opens the physical connection with NXP-NFCC and creates the required client thread for operation.
Arguments	None
Return Value	NFCSTATUS_SUCCESS if successful, Otherwise NFCSTATUS_FAILED.

API	void phNxpNciHal_TestMode_close (void)
Description	This function has to be called to close the NFCC interface and free all resources.
Arguments	None
Return Value	None

API	NFCSTATUS phNxpNciHal_SwpTest (uint8_t swp_line)
Description	This function can be used to check the SWP line as specified in "SWP Self Test" section in user manual of NXP-NFCC.
Arguments	SWP line number (1 or 2).
Return Value	NFCSTATUS_SUCCESS if successful, Otherwise NFCSTATUS_FAILED.

API	NFCSTATUS phNxpNciHal_PrbsTestStart (phNxpNfc_PrbsType_t prbs_type, phNxpNfc_PrbsHwType_t hw_prbs_type, phNxpNfc_Tech_t tech, phNxpNfc_Bitrate_t bitrate)
Description	This function can be used to test RF generation for RF technology and bit rate.
Arguments	PRBS, RF technology and bit rate as explained below. PRBS Type: 0x00 - FW software would generate the PRBS 0x01 - Hardware would generate the PRBS H/W PRBS Type: Only valid if PRBS Type 0x01 RF Technology: 0x00 - Type A 0x01 - Type B 0x02 - Type F Bitrate: 0x00 - 106 kbps (Type A,B) 0x01 - 212 kbps (Type A,B& F)

	0x02 - 424 kbps (Type A,B & F) 0x03 - 848 kbps (Type A,B)
Return Value	NFCSTATUS_SUCCESS if RF generation successful, Otherwise NFCSTATUS_FAILED.

API	NFCSTATUS phNxpNciHal_PrbsTestStop(void)
Description	This function has to be called to stop RF generation for RF technology test started by phNxpNciHal_PrbsTestStart.
Arguments	None
Return Value	NFCSTATUS_SUCCESS if operation successful, Otherwise NFCSTATUS_FAILED.

API	NFCSTATUS phNxpNciHal_AntennaSelfTest(phAntenna_St_Resp_t * phAntenna_St_Resp)
Description	This function can be used to check the Antenna's discrete component connections as specified in “Test Mode” section in user manual of NXP-NFCC.

Arguments	User can provide the reference and tolerance values from Selftest application to MW to determine the success/failure in a form of structure as below:
	`typedef struct phAntenna_St_Resp`
	`{`
	`uint16_t wTxdoRawValue; /* Txdo Raw Value*/`
	`uint16_t wTxdoMeasuredRangeMin; /*Txdo Measured Range Max */`
	`uint16_t wTxdoMeasuredRangeMax; /*Txdo Measured Range Min */`
	`uint16_t wTxdoMeasuredTolerance; /*Txdo Measured Range Tolerance */`
	`uint16_t wAgcValue; /*Agc Min Value*/`
	`uint16_t wAgcValueTolerance; /*Txdo Measured Range*/`
	`uint16_t wAgcValuewithfixedNFCLD; /*Agc Value with Fixed NFCLD Max */`
	`uint16_t wAgcValuewithfixedNFCLDTolerance; /*Agc Value with Fixed NFCLD Tolerance */`
	`uint16_t wAgcDifferentialWithOpen1; /*Agc Differential With Open 1*/`
	`uint16_t wAgcDifferentialWithOpenTolerance1; /*Agc Differential With Open Tolerance 1*/`
	`uint16_t wAgcDifferentialWithOpen2; /*Agc Differential With Open 2*/`
	`uint16_t wAgcDifferentialWithOpenTolerance2; /*Agc Differential With Open Tolerance 2*/`
	`}phAntenna_St_Resp_t;`

Return Value	NFCSTATUS_SUCCESS if operation successful, Otherwise NFCSTATUS_FAILED.

Note: The antenna selftest is to provide measurement results. The actual validation is done by comparing the measurement results returned from antenna self-test CMD, against some range of values provided by the application. The Range of values are not static, but change depending on the customer platform.

API	**NFCSTATUS phNxpNciHal_RfFieldTest (uint8_t on)**
Description	This function performs RF filed test.
Arguments	0- RF field off tests, 1- RF field on tests.
Return Value	NFCSTATUS_SUCCESS if operation successful, Otherwise NFCSTATUS_FAILED.

API	**NFCSTATUS phNxpNciHal_DownloadPinTest (void)**
Description	This function can be used to validate the FW download pin connection.
Arguments	None
Return Value	NFCSTATUS_SUCCESS if operation successful, Otherwise NFCSTATUS_FAILED.

9. DTA APK User Manual

9.1 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).

9.2 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).

9.3 Architecture of NFC DTA APK

Figure 1 shows the architecture of NFC DTA APK.

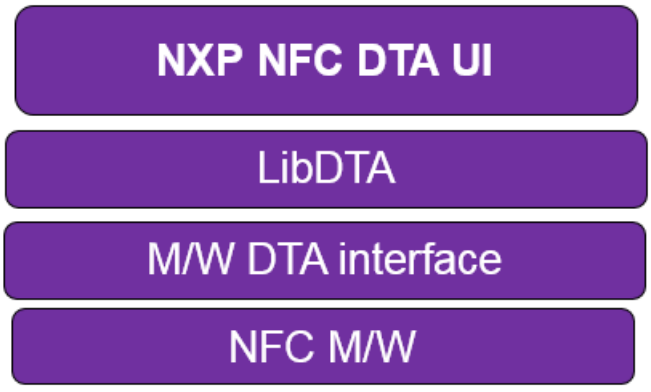


Figure 1: NFC DTA APK Architecture

9.3.1 NFC DTA supported Features:

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

9.3.2 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.

9.4 NFC DTA APK setup

DTA APK and binaries can be downloaded from NXPAndroidDTA project in GitHub using the checkout command below:

```
git clone https://github.com/NXPNFCProject/NXPAndroidDTA.git
```

The repository contains the 32bit and 64bit 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.

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

- 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.

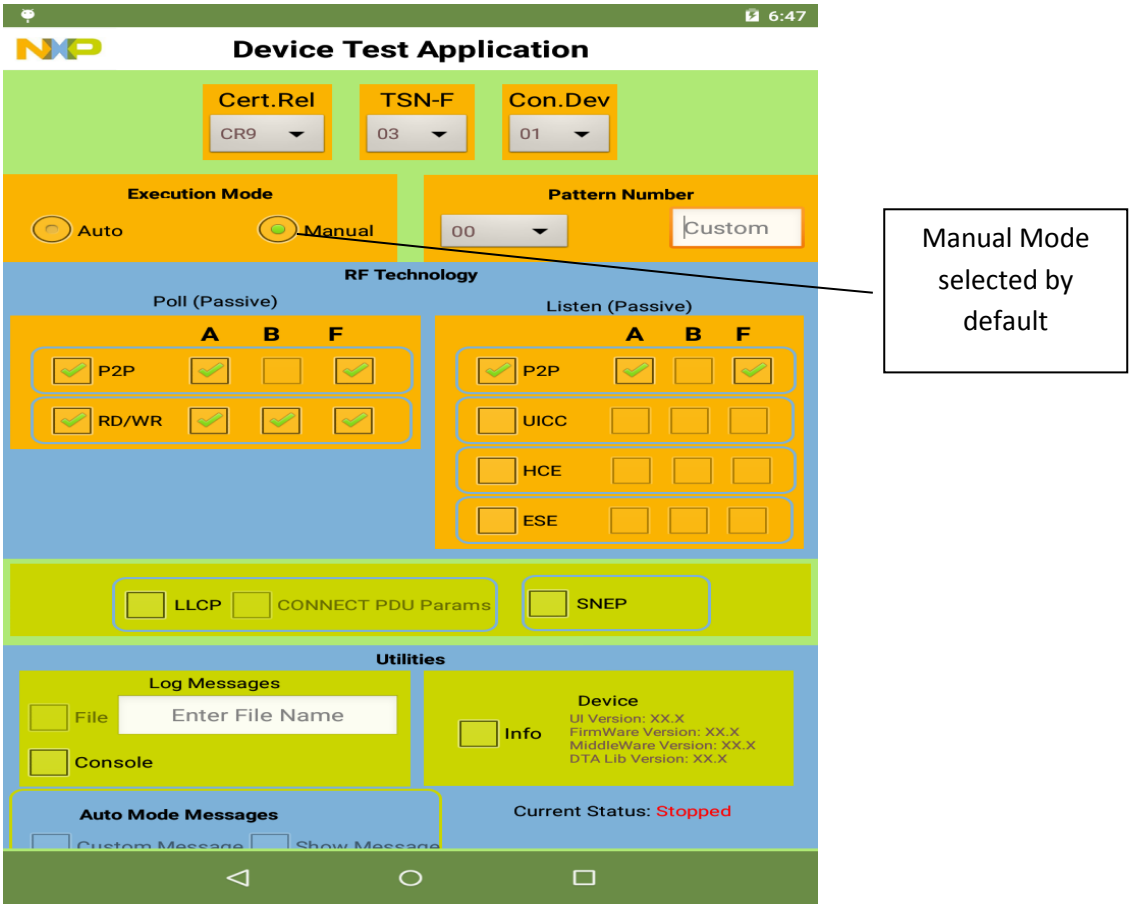
  Settings -> Display -> Sleep -> select 30 minutes.

9.5 DTA APK Menu selection

9.5.1 NXP_DTA_UI_SCR_SCENERIO_01: Default screen

The default screen is loaded as soon as the application is launched. By default certification release CR8 with Time Slot Number(TSN-F) for NFC-F technology with value 03, 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.

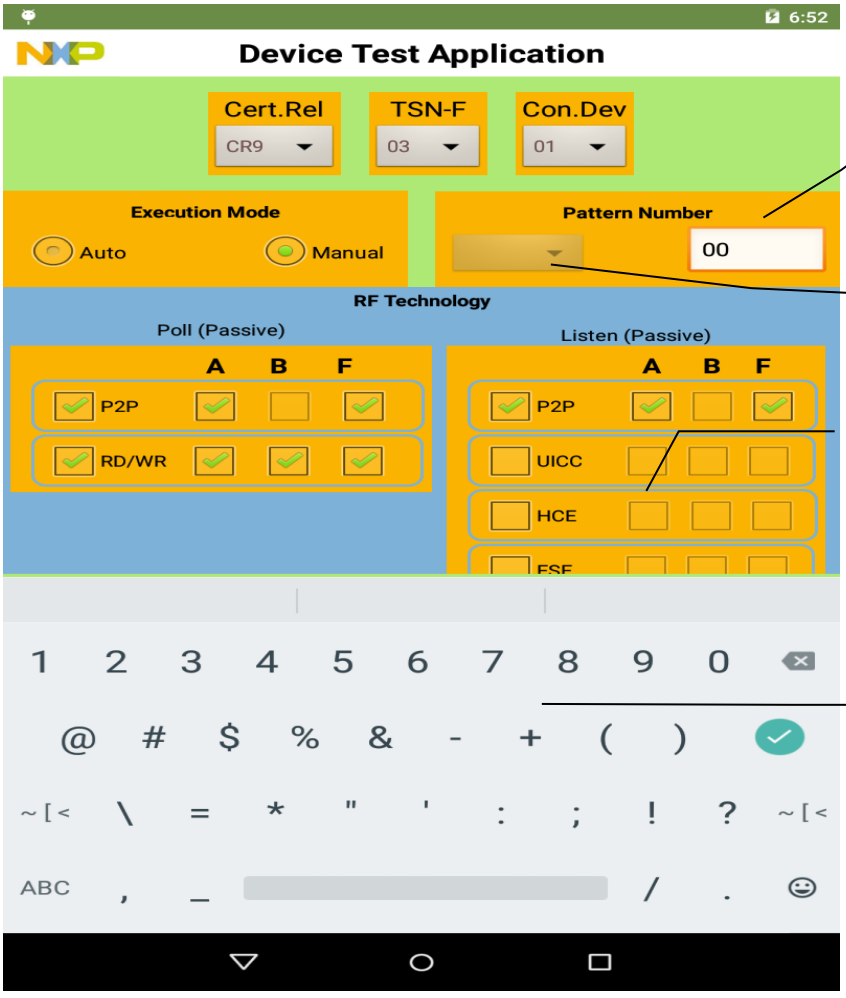


9.5.2 NXP_DTA_UI_SCR_SCENERIO_02: Selections in Manual Mode

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

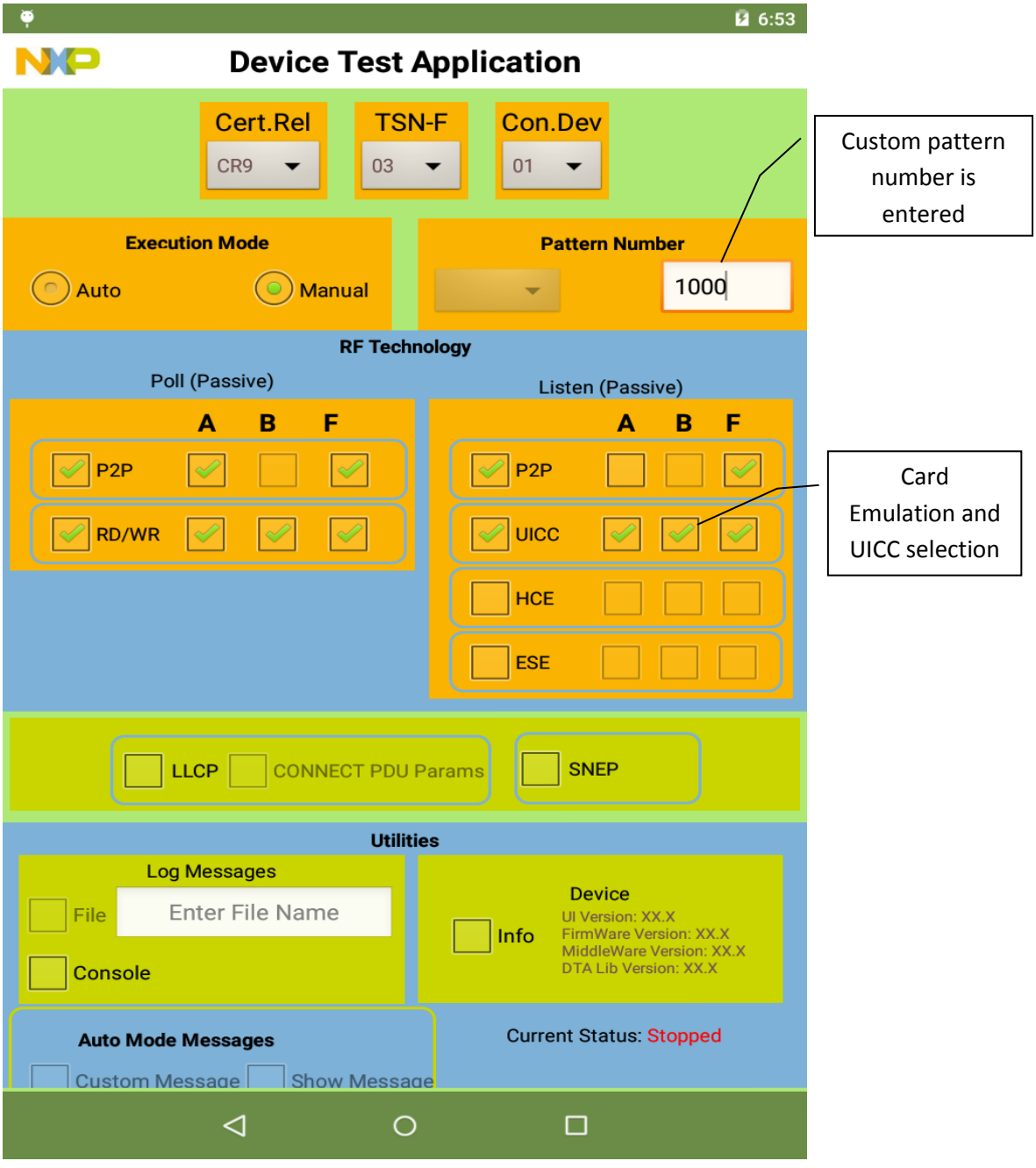
The custom pattern number is entered as 0000. Multi pattern number if selected will get cleared. Need to enter hexadecimal pattern number without the prefix 0x. Only 0000 to ffff is allowed to enter. Other entry will show pop-up message as shown in the. Maximum number of bytes allowed is only 4. As soon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



9.5.3 NXP_DTA_UI_SCR_SCENERIO_03: Analog Selection in Manual Mode

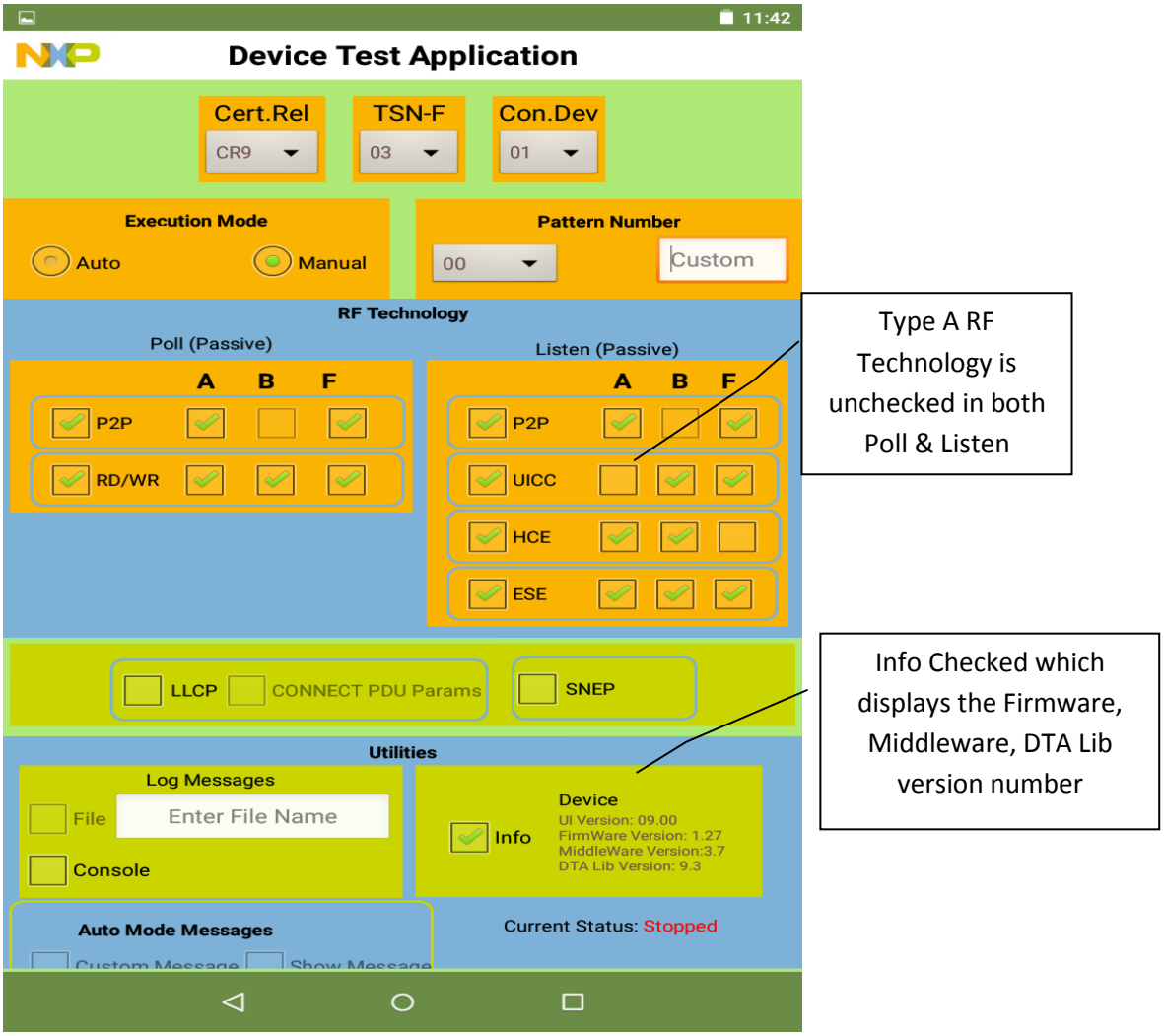
First select the CE mode with UICC. The custom pattern number is entered as 1000 and press RUN button then the application will start running in manual mode. The current status will be changed to first Running and the text color is in green. Now, all the selections are disabled.



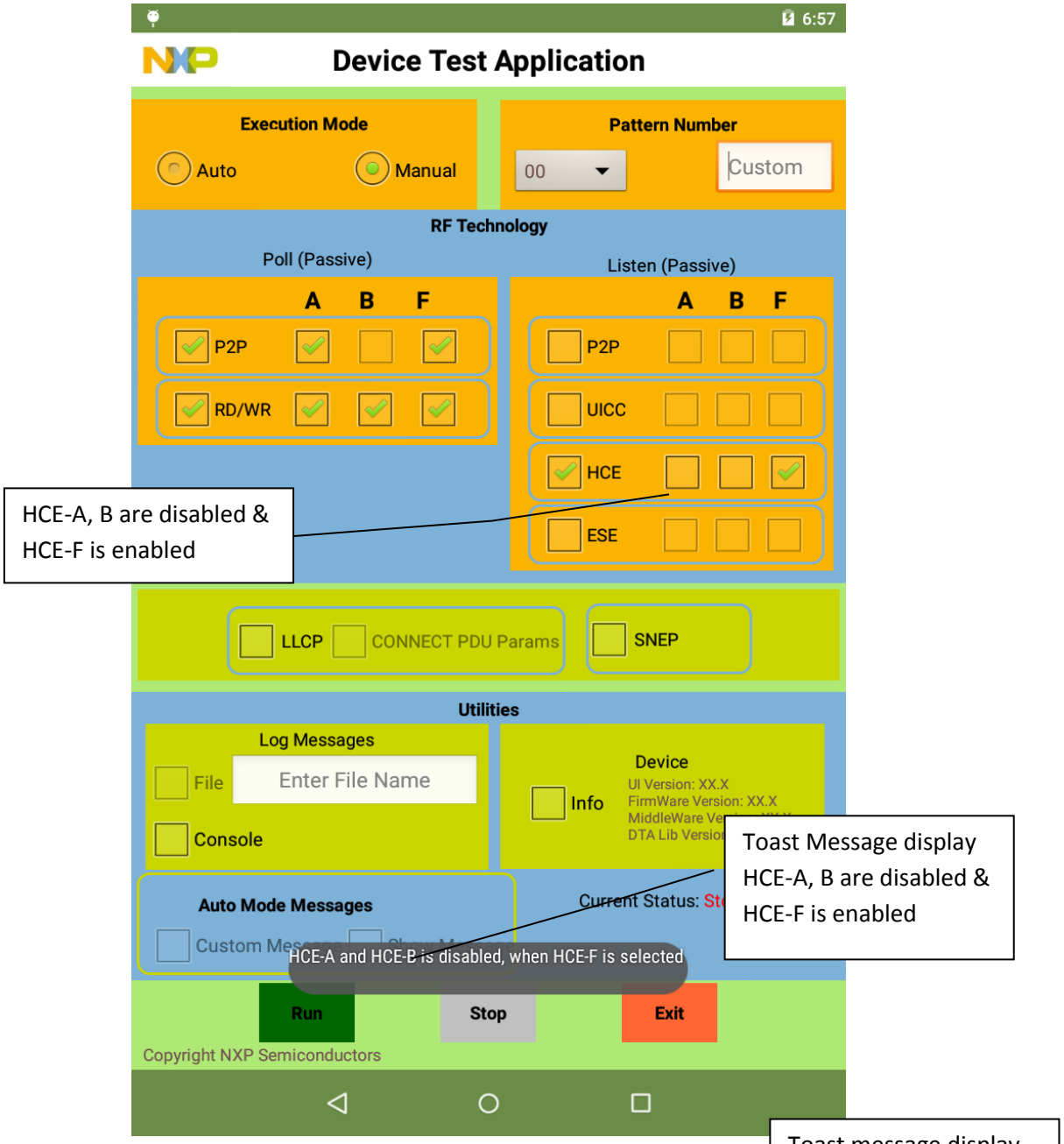
9.5.4 NXP_DTA_UI_SCR_SCENERIO_04: De-selections in Manual Mode

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.

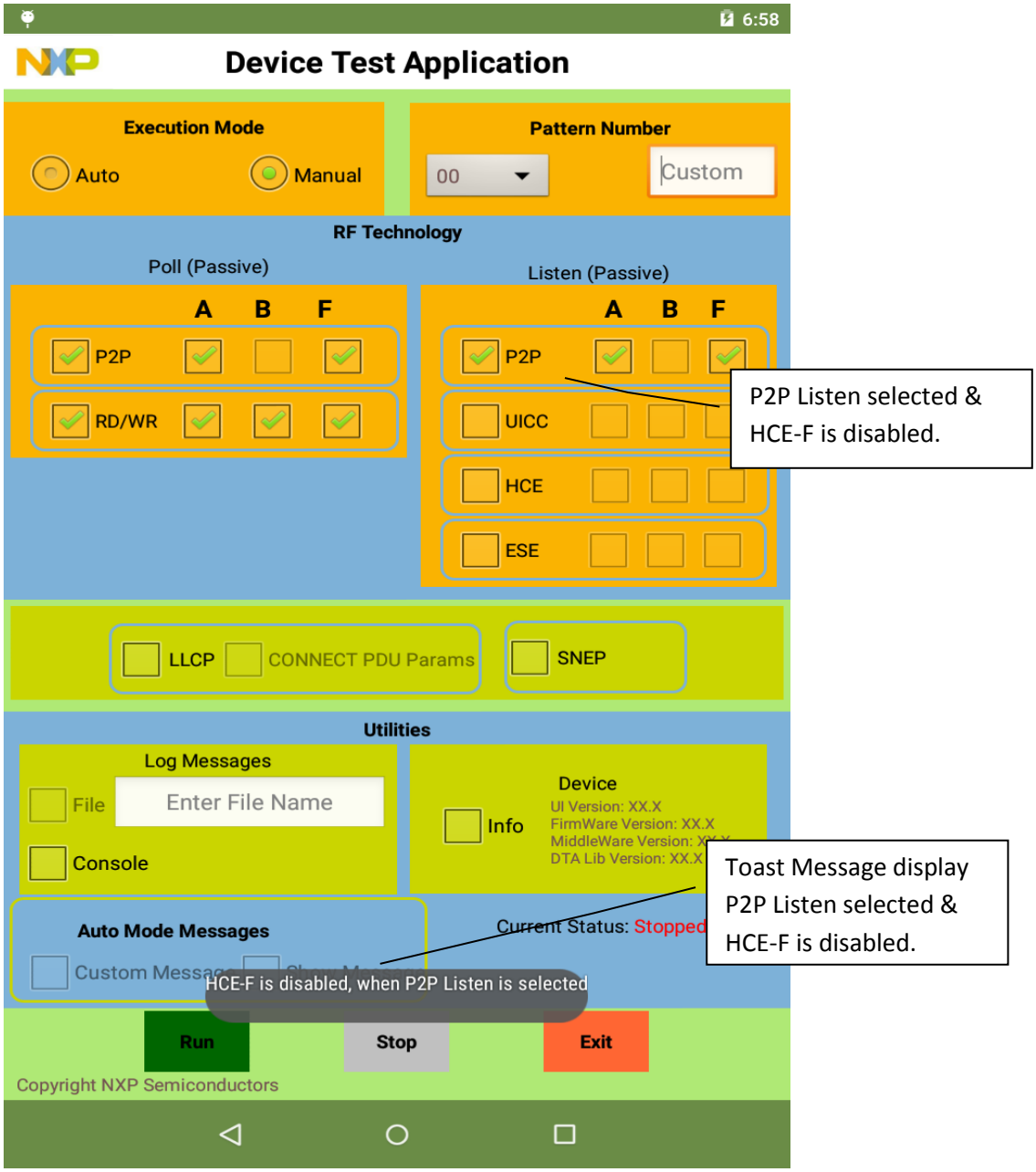


9.5.5 NXP_DTA_UI_SCR_SCENERIO_05: UI toast messages for HCE NFC-F:

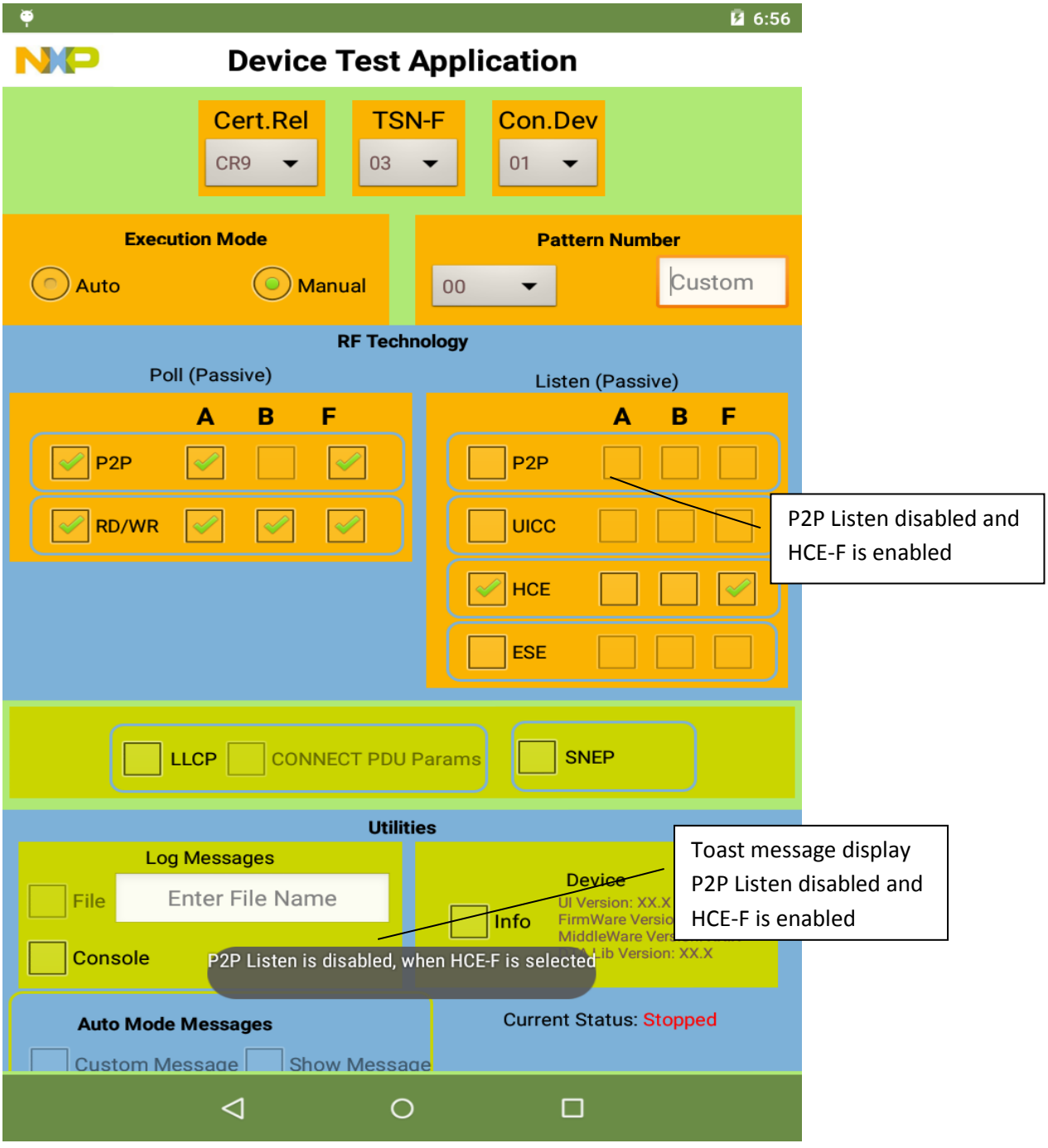


To run the Listen mode HCE NFC-F test case need to disable HCE NFC
enable HCE NFC F technology in Listen Mode.

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.

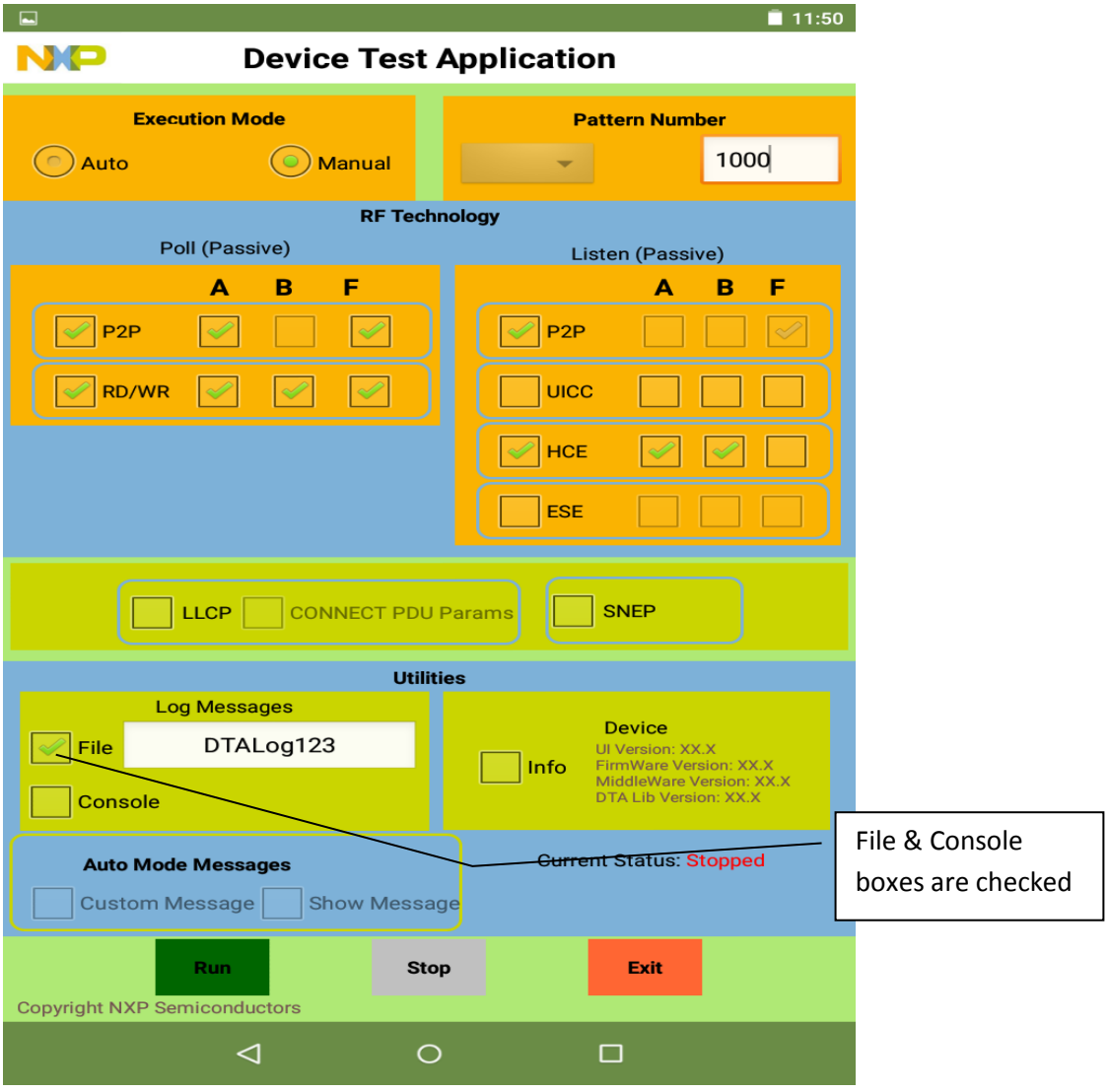


9.5.6 NXP_DTA_UI_SCR_SCENERIO_05: Device Info and Log Messages

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.

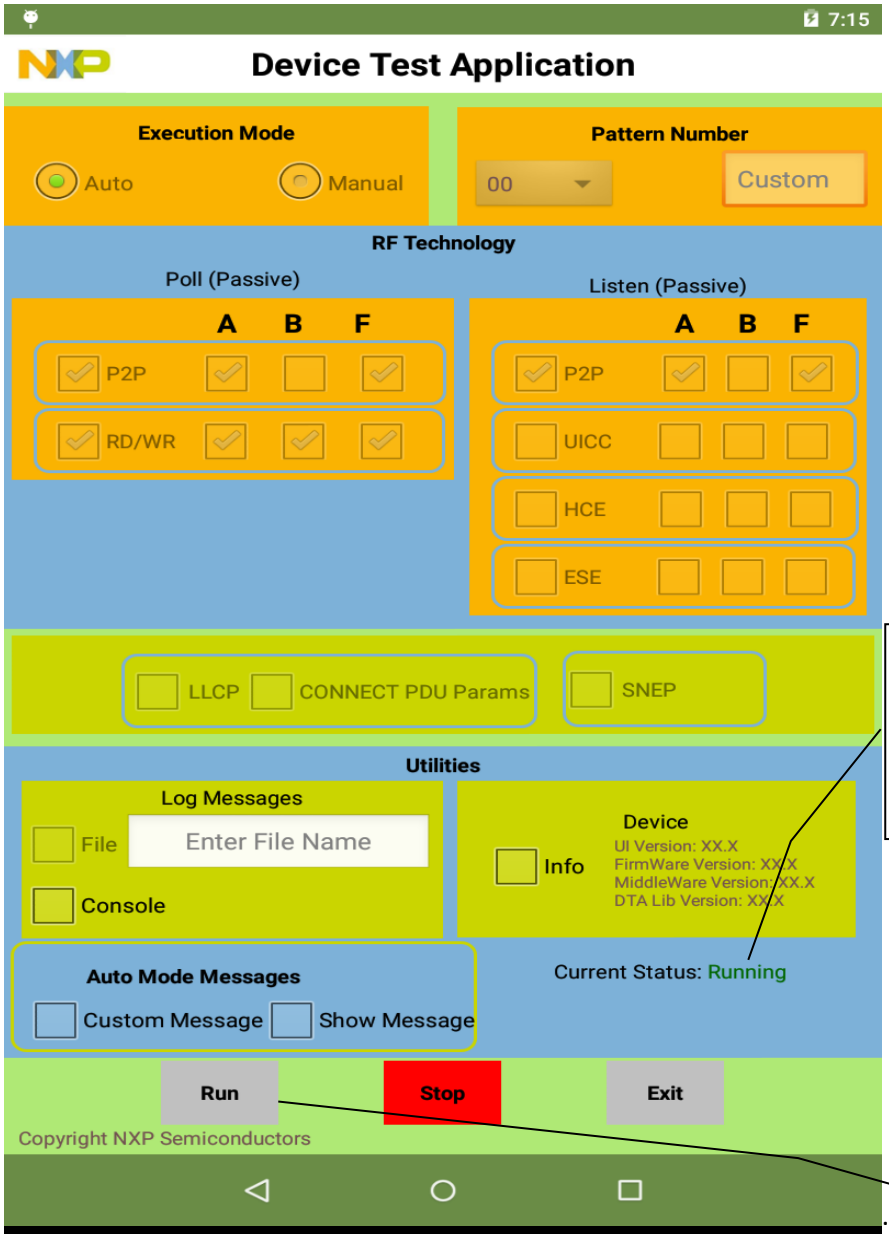
Also the user has checked the console box to see the system message logs on a separate console. The user has entered the file name and checked the File check box to write system log messages on to the SD card. By default the system logs will be stored in the directory “/sdcard/nxpdtalog/”. User need 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.



9.5.7 NXP_DTA_UI_SCR_SCENERIO_07: Running in Manual Mode

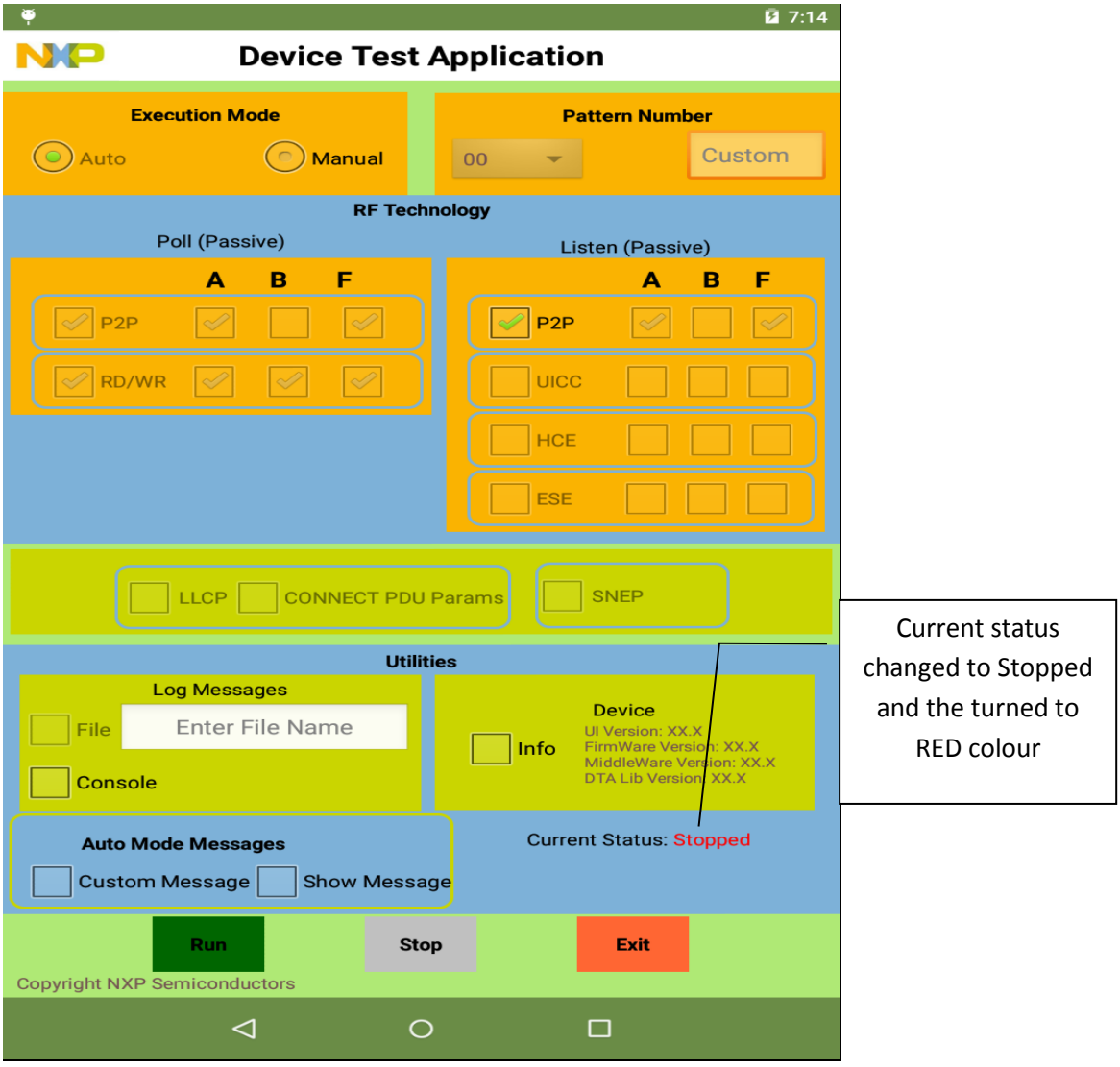
IN this screen, the user press RUN button then the application will start running in manual mode. The current status will be changed to Running and the text color is in green. During running, no other selections are allowed.

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



9.5.8 NXP_DTA_UI_SCR_SCENERIO_08: Stopping in Manual Mode

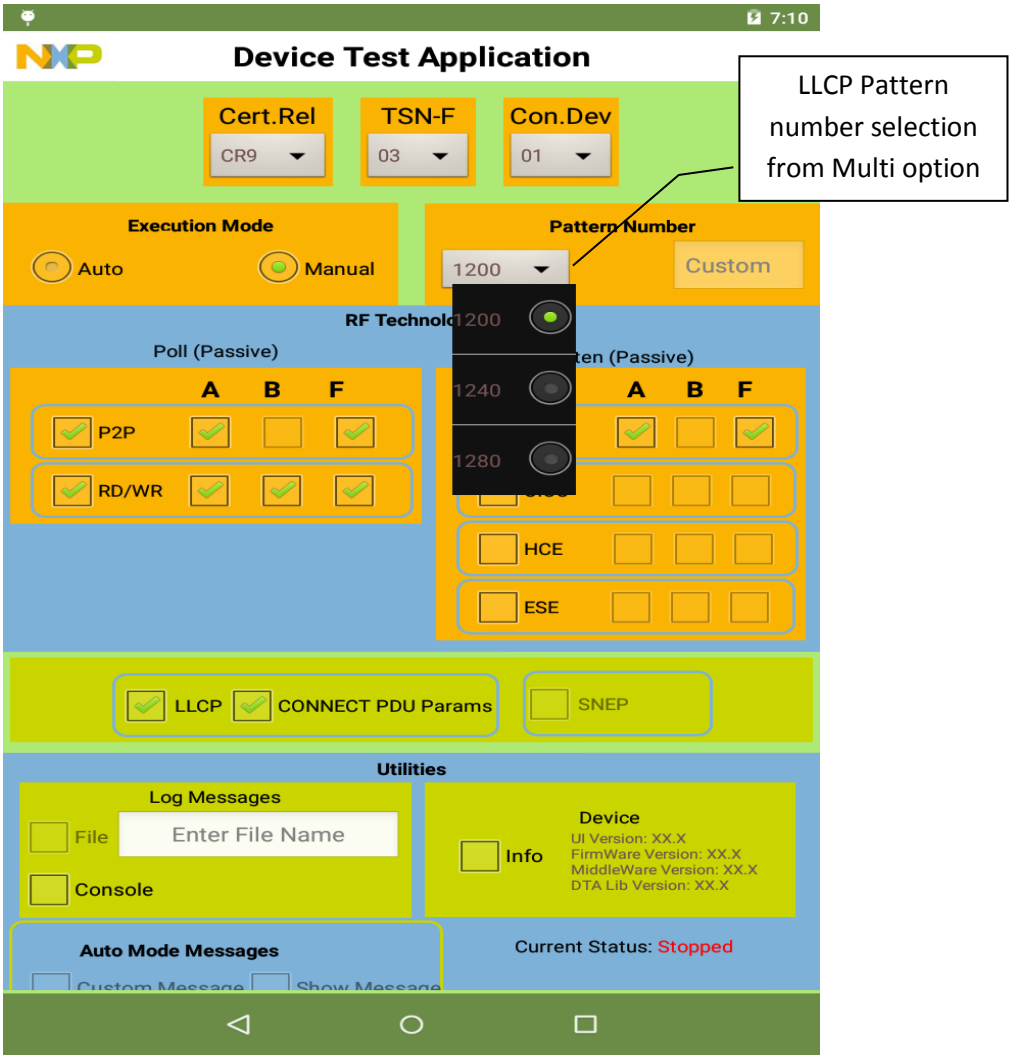
In the previous screen the user pressed Stop button to stop the application. Now the application has stopped as shown in the current screen. The current status is changed to Stopped and is in red color. Now all the selections are cleared (and will remain in manual mode) and enabled for selection except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



9.5.9 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

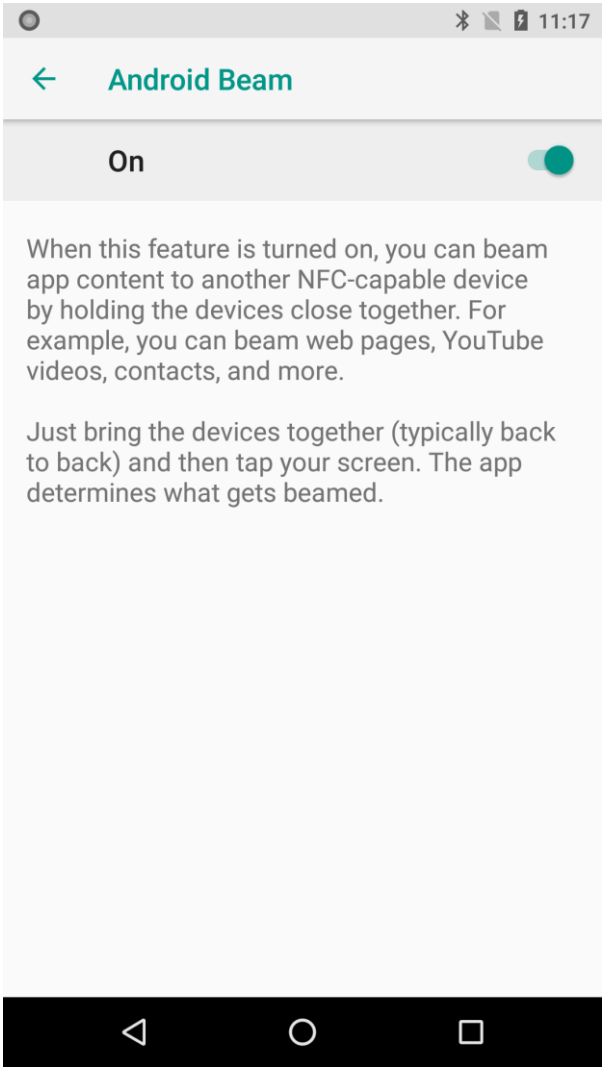
application will start running in manual mode. The current status will be changed from stopped to Running and the text color is in green. Now, all the selections are disabled. Parameters in CONNECT PDU are enabled by default when LLCP is selected. Disable this option if IUT is not required to send parameters in CONNECT PDU.

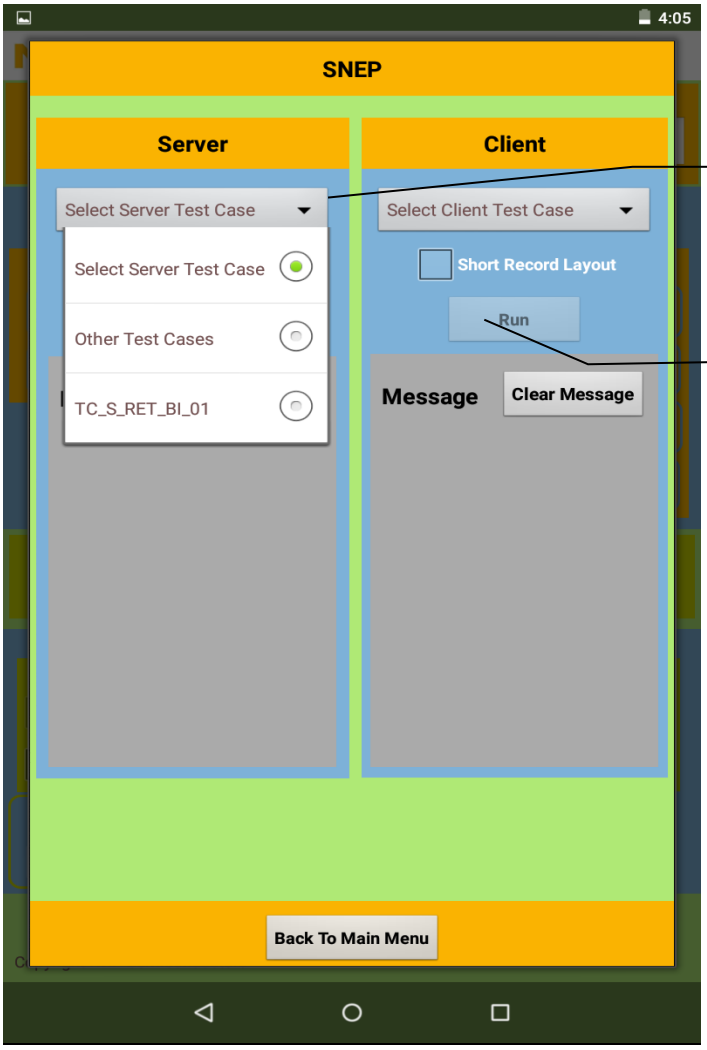


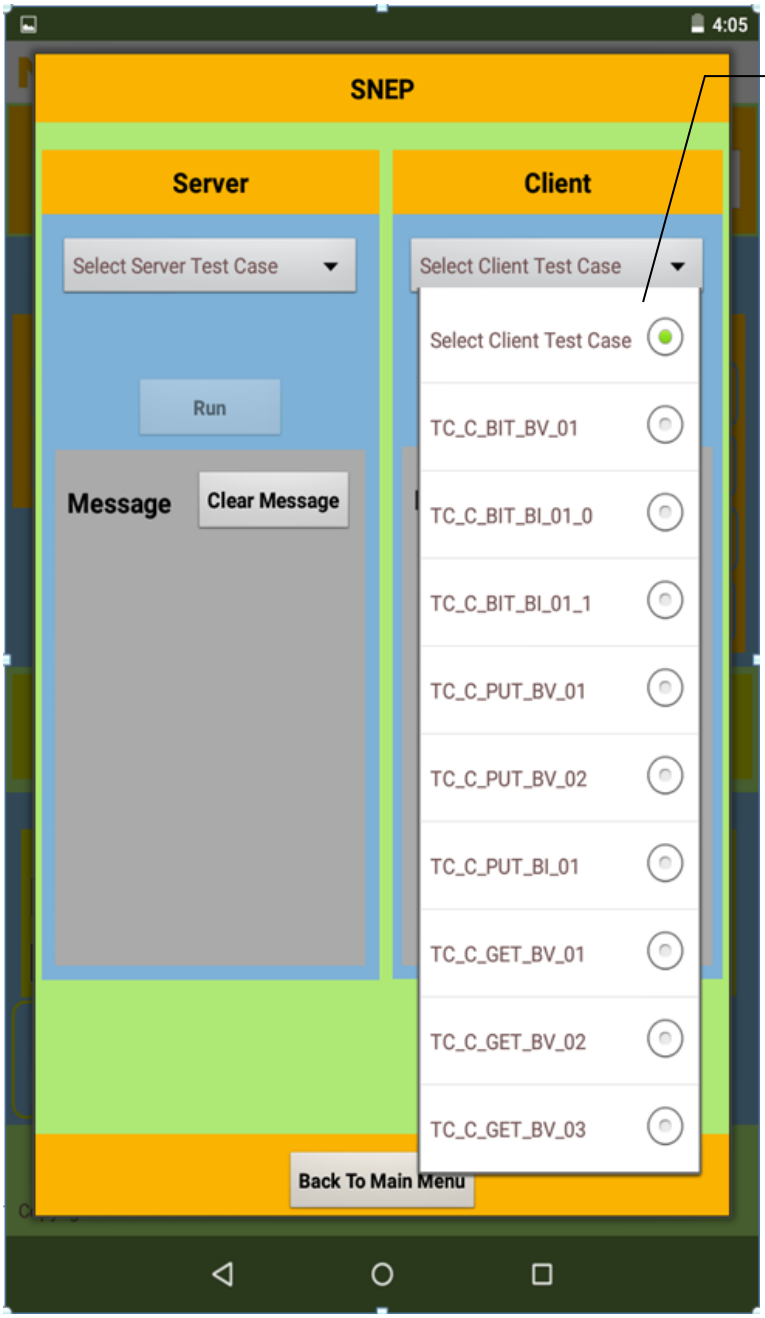
9.5.10 NXP_DTA_UI_SCR_SCENERIO_10: SNEP

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.

10. Legal information

10.1 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.

10.2 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.

10.3 Licenses

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

10.4 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>

10.5 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.

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.

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

Date of release: 13 September 20177

Document identifier: AN11762-NXP NFC Setup Guide