



# Open NFC - Security Stack and OTA Examples

Document Type:	Application Notes
Reference:	APN_NFC_1109-275 Version 0.1 (11181)
Release Date:	Sept. 12, 2011
File Name:	APN_NFC_1109-275 Open NFC - Security Stack and OTA Examples v0.1.pdf
Security Level:	General Business Use

## Disclaimer

This document is licensed under the Creative Commons Attribution 3.0 license (<http://creativecommons.org/licenses/by/3.0/>). (You may use the content of this document in any way that is consistent with this license and if you give proper attribution (<http://www.open-nfc.org/license.html#attribution>)).

Copyright © 2011 Inside Secure

Open NFC and the Open NFC logo are trademarks or registered trademarks of Inside Secure.

Other brand, product and company names mentioned herein may be trademarks, registered trademarks or trade names of their respective owners.

## History

Version	Date	Comments
0.1	Sept. 12, 2011	First Draft

## Summary of Contents

<b>1</b>	<b>Introduction.....</b>	<b>5</b>
<b>2</b>	<b>The OTA demonstration.....</b>	<b>6</b>
2.1	Architecture.....	6
2.2	The OTA Server.....	6
<b>3</b>	<b>The OTA Client.....</b>	<b>8</b>
<b>4</b>	<b>The SecStack demonstration.....</b>	<b>9</b>
4.1	Sample ACL.....	9
4.1.1	ACL #1 (acl_secstackdemo_allow.xml).....	9
4.1.2	ACL #2 (acl_secstackdemo_forbid.xml).....	10
4.2	What SecStack Demo does.....	10
4.3	Running tests.....	10

## 1 Introduction

Several examples of applications are included to demonstrate the functionalities of the Open NFC security stack.

The first example demonstrates how to create an OTA client server to perform the provisioning of the security policy for any Secure Element (embedded or UICC).

The second example shows the protection provided by the Security Stack depending on the content of the policy files.

## 2 The OTA demonstration

The purpose of this demonstration is to show the loading of an ACL into one of the Secure Elements (embedded or UICC).

Note that the code provided for this example is just an example of implementation of an OTA client/server. It is not a commercial grade product dedicated to an actual deployment.

### 2.1 Architecture

The demo uses a client/server architecture where:

- The OTA Server is a PC application that waits for incoming connections on TCP/IP.
- The OTA Client is an Android application connected to the OTA server and communicating with the Secure Element.

The communication between the client and the server uses a TCP socket. The default port is 2910 but it may be changed in both the client and the server.

Server side (PC)

```
My Computer (Server side) - java -jar OTAServer.jar acl.bin
MyComputer>java -jar OTAServer.jar acl.bin
OTA server: Press any key to exit.
Accepted connection from /172.17.10.4:54730
Starting ACL updating...
C-APDU: 80 50 00 00 08 61 06 30 45 78 DB 70 02
R-APDU: 00 00 10 23 DF C5 18 80 21 17 FF 02 00 22 9F
D1 49 AD F7 90 00
C-APDU: 84 82 01 00 10 46 9A A5 2E 73 B6 59 D3 6B 7C
R-APDU: 90 00
C-APDU: 04 A4 02 00 0A 50 36 C3 8E 53 BD 8C EC C5 1F
R-APDU: 80 02 04 00 90 00
C-APDU: 04 D6 00 00 76 00 82 03 FC F0 69 6E 63 6C 53
30 0A 04 02 50 36 02 01 39 80 01 2A 30 16 04 08 A0 0
02 50 36 02 01 63 80 01 2A 00 54 30 28 A0 16 81 14
5 85 A2 0D 6E AB B5 A6 EB C6 9C A1 0E A0 0C 04 04 90
30 28 A0 16 81 14 71 FB 36 14 DE 8D 1D 41 4C C4 8F 1
R-APDU: 90 00
C-APDU: 04 D6 00 6E 76 90 81 BC 43 25 85 A2 0D 6E AF
04 04 00 CA DF 7C 04 04 FC FF FF FF 00 82 03 6F 00 0
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
R-APDU: 90 00
C-APDU: 04 D6 00 DC 76 00 00 00 00 00 00 00 00 00 00
```

Client side (Android)

```
OTA Demonstration
Connected to the SEService
Available SE:
- SLE97-144SE (present)
- UICC (absent)

Installing ACL...
Using SE in reader SLE97-144SE
Opening session with SE...
Opening logical channel...
Connecting to the OTA Server...
Connected
C-APDU: 80 50 00 00 08 61 06 30 45 78 DB 70 02
R-APDU: 00 00 10 23 DF C5 18 80 21 17 FF 02 00
22 9E AC E4 B9 93 70 3B 27 4F 90 D1 49 AD F7
90 00
C-APDU: 84 82 01 00 10 46 9A A5 2E 73 B6 59
D3 6B 7C F1 D0 8A 55 B0 19
R-APDU: 90 00
C-APDU: 04 A4 02 00 0A 50 36 C3 8E 53 BD 8C
EC C5 1F
R-APDU: 80 02 04 00 90 00
C-APDU: 04 D6 00 00 76 00 82 03 FC F0 69 6E 63
```



### 2.2 The OTA Server

The OTA Server reads the ACL stored in the binary file passed on its command line and then waits for incoming connections. Once a connection is established, the OTA Server generates the APDU command flow required to install the ACL onto the Secure Element: It sends each APDU command to the connected OTA client and then waits for the response APDU. As long as the response is correct (that is, typically, contains a status word of 9000), it continues until the whole ACL is transmitted. The APDU command flow consists of the following commands:

- INITIALIZE UPDATE and EXTERNAL AUTHENTICATE, to establish a GlobalPlatform secure channel with the PKCS#15 application of the SE;
- UPDATE BINARY (with MAC), as many times as needed, to send the contents of the ACL;
- UPDATE BINARY (with MAC) to update the contents of the “Last Update” file;
- CLOSE SESSION (with MAC) to close the secure channel.

The OTA Server does not send the SELECT[application] command to select the PKCS#15 application in the SE. This is done by the OTA Client.

For the mutual authentication with the SE, the OTA Server assumes that the GlobalPlatform secure channel protocol SCP '02' option i=15 is used.

The OTA Server command line syntax is as follows:

```
MyComputer>java -jar OTAServer.jar
Usage: java -jar otaserver.jar [options] <acl.bin>
where options include:
  -?          Print usage and exit
  -k <key>    Specify master key (default: 404142...4F)
  -p <port>   Specify TCP port (default: 2910)
  -t <timeout> Specify connection timeout (in seconds)
```

The only mandatory parameter is the binary file containing the ACL. This file may be generated by the acngen tool from an XML representation of the ACL.

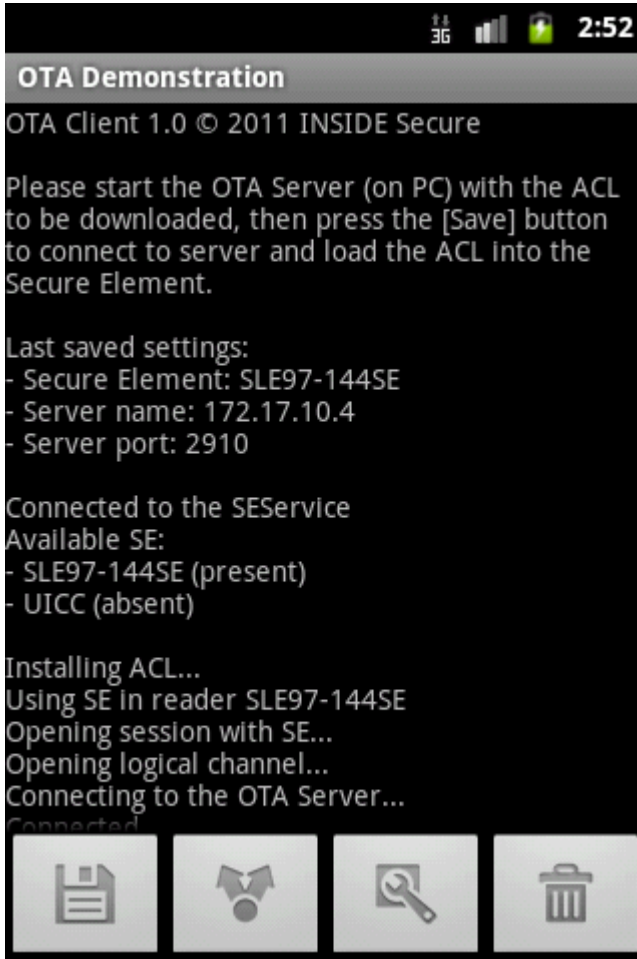
The -k parameter contains the hexadecimal value of the 2-key Triple DES master key used to perform mutual authentication. Its default value is "404142434445464748494A4B4C4D4E4F".

The -p parameter contains the TCP port to be used.

The -t parameter contains the time after which the OTA server stops if there is no incoming connection. The default is 10 minutes. The value 0 is interpreted as an infinite timeout.

### 3 The OTA Client

The OTA client is an Android application with the following user interface:



Meaning of the icons:



Start downloading the ACL on the current SE from the currently selected OTA server



Select the Secure Element on which the ACL is going to be installed.



Change the settings of the current OTA Server.



Clear the log window.

When the OTA Client starts, it reads its settings (last used Secure Element, and last used OTA Server name and port) from its persistent store. It also connects to the SE service and lists the available SE.

When the user asks for downloading and installing the ACL, the OTA Client connects to the SE, opens a logical channel and selects the PKCS#15 application on it. It then establishes a connection with the OTA server. All APDU commands sent by the OTA Server are transmitted as is to the SE.

**IMPORTANT NOTE:** In order to allow the OTA Client application to communicate with the PKCS#15 application of the SE, it must have been signed with the certificate of the default principal.

## 4 The SecStack demonstration

Once an ACL have been installed, the "SecStack demo" Android application can be used to check that some APDU commands are accepted and all others are rejected.

**IMPORTANT NOTE:** The SecStack Demo application must not be signed with the certificate of the default principal.

### 4.1 Sample ACL

The demo uses the two ACL file described below. They only refer to the ISD and Jupiter applets that are likely always present on the SE stacked the NFC Controller, so that there is no need to load and install any Java Card applets on the SE to run the tests.

#### 4.1.1 ACL #1 (acl\_secstackdemo\_allow.xml)

This ACL allows the SecStack Demo to:

- Select the Jupiter applet and send it the GET DATA (tag='0102') command
- Select the ISD applet and send it the GET DATA (tag='DF7C') command

All other APDU commands are rejected.

```
<?xml version="1.0"?>
<acl format="1.0">

  <!-- ACIE associated with the Jupiter SE application -->
  <acie ace="idJupiter">
    <aid>0xF0 0x69 0x6E 0x63 0x6C 0x53 0x45 0x53 0x65 0x74 0x74 0x69 0x6E 0x67
    0x73</aid>
  </acie>

  <ace id="idJupiter">
    <principals>
      <principal type="apk">bin\SecStackDemo.apk</principal>
    </principals>
    <permissions>
      <!-- Allow GET DATA[0102] commands -->
      <permission type="APDU">
        <header>0x90 0xCA 0x01 0x02</header>
        <mask>0xFC 0xFF 0xFF 0xFF</mask>
      </permission>
    </permissions>
  </ace>

  <!-- ACIE associated with the ISD application -->
  <acie ace="idISD">
    <aid>0xA0 0x00 0x00 0x00 0x03 0x00 0x00 0x00</aid>
  </acie>

  <ace id="idISD">
    <principals>
      <principal type="apk">bin\SecStackDemo.apk</principal>
    </principals>
    <permissions>
      <!-- Allow GET DATA[DF7C] commands -->
```

```
<permission type="APDU">
  <header>0x00 0xCA 0xDF 0x7C</header>
  <mask>0xFC 0xFF 0xFF 0xFF</mask>
</permission>
</permissions>
</ace>

</acl>
```

## 4.1.2 ACL #2 (acl\_secstackdemo\_forbid.xml)

This ACL allows the SecStack Demo to only select the Jupiter applet. All other APDU commands are rejected.

```
<?xml version="1.0"?>
<acl format="1.0">

  <!-- ACIE associated with the Jupiter SE application -->
  <acie ace="idJupiter">
    <aid>0xF0 0x69 0x6E 0x63 0x6C 0x53 0x45 0x53 0x65 0x74 0x74 0x69 0x6E 0x67
    0x73</aid>
  </acie>

  <ace id="idJupiter">
    <principals>
      <principal type="apk">bin\SecStackDemo.apk</principal>
    </principals>
    <permissions>
      <!-- FORBID ALL APDU COMMANDS (but the SELECT[application] command) -->
      <permission type="APDU">
        <header>0xFF 0xFF 0xFF 0xFF</header>
        <mask>0x00 0x00 0x00 0x00</mask>
      </permission>
    </permissions>
  </ace>

</acl>
```

## 4.2 What SecStack Demo does

When the "Run Test..." button is pressed, the application sends the following APDU commands in sequence to the SE. The results expected for the two ACL given above are also shown.

APDU command	ACL #1	ACL #2
MANAGE CHANNEL [open]	Success	Success
SELECT[application] with ISD AID	Success	Failure
GET DATA 'DF7C'	Success	Failure
MANAGE CHANNEL [close]	Success	Success
MANAGE CHANNEL [open]	Success	Success
SELECT[application] with Jupiter AID	Success	Success
GET DATA '0102'	Success	Failure
MANAGE CHANNEL [close]	Success	Success

Table 1: APDU commands sent and expected results

## 4.3 Running tests

To check that the ACL apply, please proceed as follows:

# Open NFC - Security Stack and OTA Examples

General Business Use

Page : 11/11

Date : Sept. 12, 2011

Ref. : APN\_NFC\_1109-275 v0.1(11181)

---

1. Run the OTA Application (both server and client) and load the `acl_secstackdemo_allow.bin` ACL into the SE.
2. Run the SecStack Demo application and check that obtained results are those given in Table 1 just above (all APDU commands must pass)
3. Run the OTA Application (both server and client) and load the `acl_secstackdemo_forbid.bin` ACL into the SE
4. Run the SecStack Demo application and check that obtained results are those given in Table 1 just above (some APDU commands must be rejected)