



Open NFC - Usage of the SE API with SecuRead 1.0

Document Type:	Application Notes
Reference:	APN_NFC_1011-231 Version 1.3 (11510)
Release Date:	Oct. 3, 2011
File Name:	APN_NFC_1011-231 Open NFC - Usage of the SE API with SecuRead 1.0 v1.3.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 © 2010-2011 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	Nov. 17, 2010	First Draft
1.0	Dec. 24, 2010	Release for Open NFC 4.2
1.1	Jan. 2, 2011	Adding precisions on the power configuration in battery-off mode. Updating the document license.
1.2	June 6, 2011	Describing the new automatic implementation in Open NFC 4.3. Removing the specific Jupiter API no longer present in Open NFC 4.3.
1.3	Oct. 3, 2011	Updating the text with the new Secure Element API of Open NFC 4.3.1.

Summary of Contents

1	Introduction	5
2	Secure Element Policy	6
2.1	Supported Protocols	6
2.2	Synchronization of the NFC Controller Policy	6
2.3	Setting the Policy during Boot Time	7
2.4	Setting the Policy at Runtime.....	7
2.5	Automatic Configuration of the Power in Battery-Off Mode	8
3	End of Transaction Detection	9
3.1	Behavior of SecuRead	9
3.2	Buit-in Transaction History in the Secure Element.....	9
3.3	Behavior of the Open NFC Functions	9
4	Applet Protection	11
5	Migration from the Last Versions of the API	12

1 Introduction

This document describes the usage of Open NFC™ API on top of a SecuRead™ 1.0 chip and specifically the Secure Element.

The audience of this document should be familiar with the Open NFC API and the generic behavior of the Secure Elements. This document describes only the specificities of the SecuRead Secure Element.

The behavior of the Secure Element of the SecuRead 1.0 chip is managed with the Open NFC Secure Element API. The main specificities of the Secure Element included in the SecuRead 1.0 chip are related to policy used to control card emulation and the detection of the AIDs used during the transactions.

References

APN_NFC_1012_519_CONFIDENTIAL_SE Configuration Applet, v1.8

APN_NFC_1012_520 Jupiter Application Notes User Interaction Mechanisms

2 Secure Element Policy

2.1 Supported Protocols

The Secure Element supports card emulation in ISO 14443 level 4 A and B, ISO 15693 for IClass emulation and ISO 14443 A level 3 for Mifare emulation.

Open NFC Constants	Secure Element Card Emulation
W_NFCC_PROTOCOL_CARD_ISO_14443_4_A	ISO 14443 level 4 A
W_NFCC_PROTOCOL_CARD_ISO_14443_4_B	ISO 14443 level 4 B
W_NFCC_PROTOCOL_CARD_ISO_15693_2	IClass emulation
W_NFCC_PROTOCOL_CARD_MIFARE_CLASSIC	Mifare emulation

The following array describes the protocol combinations supported by the Secure Element:

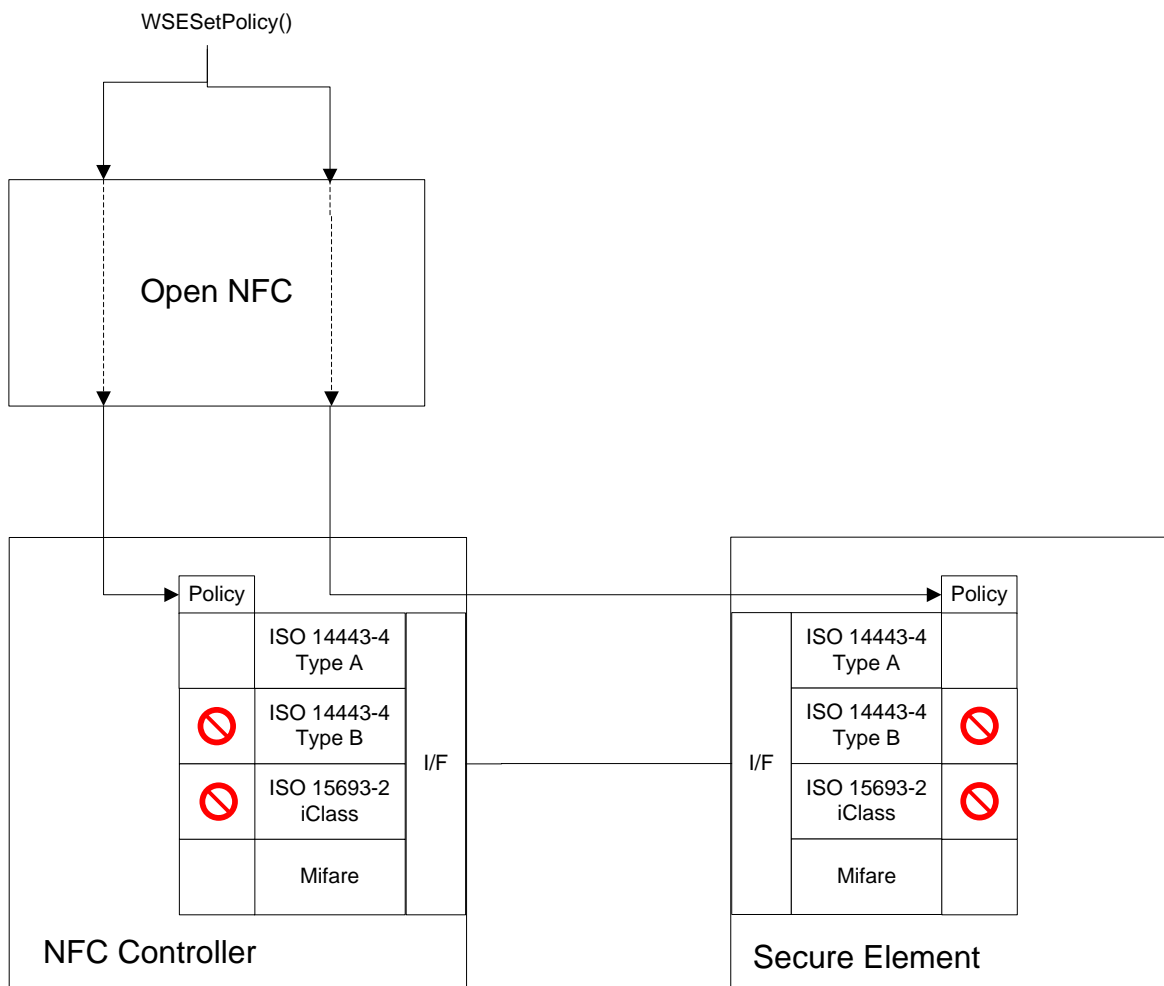
- None
- ISO 14443 A
- ISO 14443 B
- IClass
- ISO 14443 A and Mifare
- ISO 14443 A and IClass
- ISO 14443 B and IClass
- ISO 14443 A, Mifare and IClass

Using any other combination of protocols with a function of the Secure Element API will cause an error W_ERROR_FEATURE_NOT_SUPPORTED.

2.2 Synchronization of the NFC Controller Policy

The NFC Controller and the Secure Element manage two separate lists of protocols allowed for the Secure Element in card emulation. The lists are synchronously updated with the function WSESetPolicy().

To update the Secure Element list, WSESetPolicy() opens a connection with the Secure Element and exchanges APDUs to set the policy with the values currently used by the NFC Controller. Using this function may temporarily block the access to the Secure Element from the other handset applications.



2.3 Setting the Policy during Boot Time

The Secure Element stores the protocol policy in persistent memory. During the boot sequence of the handset, the Open NFC stack synchronizes the policy used by the Secure Element with the policy used by the NFC Controller.

2.4 Setting the Policy at Runtime

Each time the handset firmware calls `WSESetPolicy()` to set the NFC Controller policy, the new value is updated in the Secure Element policy.

2.5 Automatic Configuration of the Power in Battery-Off Mode

In Battery-Off mode, only one element can be powered: the Embedded Secure Element or the UICC. The selection of the powered element is automatically configured with `WSESetPolicy()`.

If the persistent policy set with `WSESetPolicy()` allows a card emulation for the embedded SE, `WSESetPolicy()` configures the NFC Controller to power on the SE in Battery-Off mode. With this configuration, the UICC is not powered in battery off mode regardless of the UICC policy. If the persistent policy set with `WSESetPolicy()` does not allow card emulation for the embedded SE, the power configuration is left unchanged.

Symmetrically, if the persistent policy set with `WSESetPolicy()` allows a card emulation for the UICC, `WSESetPolicy()` configures the NFC Controller to power on the UICC in Battery-Off mode. With this configuration, the embedded SE is not powered in battery off mode regardless of the SE policy. If the persistent policy set with `WSESetPolicy()` does not allow card emulation for the UICC, the power configuration is left unchanged.

3 End of Transaction Detection

The detection of the end-of-transaction events is a function of the Open NFC stack. The end-of-transaction event informs the handset firmware or the handset application that a transaction occurred with the Secure Element and provides the AID (application identifiers) used for the transaction.

3.1 Behavior of SecuRead

Listening to end-of-transaction events is initiated with `WSEMonitorEndOfTransaction()`. The SecuRead chip includes a APDU listener mechanism to detect any SELECT AID APDU used during the transaction and to store the AID values. At the end of the transaction the Secure Element sends a flag (SE Info Flag) to the SecuRead to indicate if some AID are stored in the transaction log of the Secure Element. So at the end of the transaction, when the field is off, the SecuRead chip sends the end-of-transaction events with the information of the SecuRead APDU listener and the SE Info Flag combined in a manner to determine that a transaction has really taken place:

When SecuRead detects a field off event, it checks its own "SELECT AID APDU" log data. When no AID is registered then he checks the last value of the SE Info Flag. When the SE Info Flag indicates that the SE has entries in the transaction history, then he signals this to the host.

The host can receive the following messages (or not receive a message):

- No message to host when no AID in SecuRead AND SE Info Flag is zero
- Message to host with no AIDs when SecuRead has detected no AID but SE Info Flag is one
- Message to host with AID = 'FF FF' (LV = '02 FF FF') when SecuRead has detected AIDs AND SE Info Flag is zero
- Message to host with detected AIDs when SecuRead has detected AIDs and SE Info Flag is one

3.2 Built-in Transaction History in the Secure Element

The SE offers the functionality to record the AID of the applet that is selected in the moment when a persistent data modification occurs in the SE. This allows detecting active applets even when they are not explicitly selected on the RF interface. The AIDs are stored in a Transaction History log.

The Mifare and iClass emulations are detected as well due to their assigned "virtual AIDs". The Mifare and iClass transactions are included in the log even if there is no persistent data modification. The virtual AIDs are the following:

- Mifare: D2 76 00 00 05 AC 00 04 03 E0 10 01 01
- iClass: D2 76 00 00 05 AC 00 04 03 E0 10 01 02

3.3 Behavior of the Open NFC Functions

The following array summarizes the behavior of the Open NFC API:

Open NFC - Usage of the SE API with SecuRead 1.0

General Business Use

Page : 10/12

Date : Oct. 3, 2011

Ref. : APN_NFC_1011-231 v1.3(11510)

Scenario	Behavior of the Open NFC API	
	Callback Function Called with the protocol value	Value returned by WSEGetTransactionAID()
Mifare transaction	W_NFCC_PROTOCOL_CARD_ISO_14443_4_A	Mifare virtual AID
iClass transaction	W_NFCC_PROTOCOL_CARD_ISO_15693_2	iClass virtual AID
ISO 14443 A or B with at least a SELECT AID APDU and a modification of the SE persistent data.	W_NFCC_PROTOCOL_CARD_ISO_14443_4_A or W_NFCC_PROTOCOL_CARD_ISO_14443_4_B	List of AIDs including (double values are removed): - The AIDs included in the SELECT AID APDUs. - The AIDs of the applications modifying the EEPROM
ISO 14443 A or B with no SELECT AID APDU but a modification of the SE persistent data.	W_NFCC_PROTOCOL_CARD_ISO_14443_4_A or W_NFCC_PROTOCOL_CARD_ISO_14443_4_B	List of the AIDs of the applications modifying the EEPROM
ISO 14443 A or B with at least a SELECT AID APDU but no modification of the SE persistent data.	Not called	Empty
Anti-collision in ISO 14443 A or B but without transaction	Not called	Empty

4 Applet Protection

Open NFC relies on an applet in the SE to set the policy or to get the transaction log. If the handset firmware includes a security stack on top of Open NFC to protect the SE access, it shall forbid any access to this applet. The applet AID to forbid is the following:

F0 69 6E 63 6C 53 45 53 65 74 74 69 6E 67 73

5 Migration from the Last Versions of the API

In the last versions of the API (4.2.x), the following functions were defined: WJupiterSEApplyPolicy(), WJupiterSEApplyPolicySync(), WJupiterSEGetAID() and WJupiterSEGetAIDSync().

WJupiterSEApplyPolicy() or WJupiterSEApplyPolicySync() had to be called when the stack was booting and just after a call to WSESetPolicy() or WSESetPolicySync(). The corresponding function is now automatically performed by the NFC stack. The calls to WJupiterSEApplyPolicy() or WJupiterSEApplyPolicySync() shall be removed.

WJupiterSEGetAID() or WJupiterSEGetAIDSync() had to be call in complement to WSEGetTransactionAID(). Then exact list of AIDs was deduced from a combination of the AID lists returned by both functions. Now WJupiterSEGetAID() returns the exact list of AIDs. The calls to WJupiterSEGetAID() or WJupiterSEGetAIDSync() shall be removed.