Archived:Nokia 6131 NFC - FAQs

Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Article for old version of Series 40.

Nokia 6131 NFC mobile phone - Generic

Where is the antenna located in Nokia 6131 NFC?

On the top part of the flip cover of the phone. It's not around the display, but slightly above it when you hold the phone upright.

Is the reading distance shorter in Nokia 6131 NFC than in Nokia 3220 NFC shell?

This depends on quite a few factors, and therefore there is no simple answer to this. There should not be a lot of difference, but real world measurements will tell the truth.

How is the power management handled in Nokia 6131 NFC and how does it compare to the Nokia 3220 NFC shell?

The power management of the 6131NFC is considerably more advanced than in the 3220 NFC shell, so in general you can expect better performance. However, it always depends on your usage patterns.

Can third parties add things to the NFC settings menu?

No. This is an S40 phone, which does not have the flexibility for native code as S60 would.

Can the MIDlets be put into a secure area so that they cannot be deleted by the user?

The user has control over the MIDlets in the phone, including the ability to delete them. (Note that this does not apply to the secure applets in the internal secure element; just the regular MIDlets.)

What target types are supported by JSR-257 implemented in Nokia 6131 NFC mobile phone?

The supported target type values are:

- TargetType.ISO14443_CARD for ISO 14443-4 compliant smart cards accessed using APDU commands
- TargetType.NDEF_TAG for a tag that contains NFC Forum formatted data
- TargetType.RFID_TAG for general RFID tags

PLEASE NOTE: Visual tags are not supported by the Nokia 6131 NFC mobile phone, nor Nokia 6131 NFC SDK.

Secure Element & Smart Card

What are the technical details of the secure element integrated in Nokia 6131 NFC?

The operating system is Giesecke & Devrient's (G&D) Sm@rtCafé Expert 3.1.

The secure element consists of Java Card area and Mifare 4K area (behaves also as Mifare 1k) for tag emulation. The Java Card area is compliant to Global Platform 2.1.1 (http://www.globalplatform.org/specificationview.asp?id=card) and compliant to Java Card 2.2.1 (http://www.oracle.com/technetwork/java/javacard/overview/index.html).

Applets in Java Card area can access the Mifare 4k area with G&D specific libraries (ExtSystem) provided by Sm@rtCafé® Professional Toolkit.

What is the memory size of the secure element?

Approximately 65 kB. The overall memory size is 72kB, however some space is required for product specific applications and mifare 4k area.

How the Nokia 6131 NFC internal Mifare 4k area or the internal Java Card area could be utilized for storing critical information?

It depends totally of the application type and technical infrastructure of the environment, where the application is going to be used. However, following things could be considered:

- Mifare 4k area is just a memory with access control, and typically it is simpler to implement.

- Java Card provides high security environment and can execute code, which means it can be used for more complex applications. Java Card supports standardized and well known security algorithms.

- Managing several applications on Mifare side is challenging.

- On Java Card side, the amount of applications is the amount, which can fit into Java Card memory.

Which third party issuers (trusted service managers) can manage the applications on the secure element?

For example Venyon, Cassis, ViVOtech.

Who will give me the keys for the secure element of my phone?

You will not receive the keys just for development purpose; just unlock the phone, use the standard keys and then change them in order to have your applications protect for developing. Always keep in mind: the secure element is not a playground.

When you receive a "locked" phone (= secure element with production keys of a trusted party) you are not able to put applications there. These keys are hold be a TSM (trusted service manager). As the secure element is a high secure memory with the phone, only an institute that is able to hold the keys in a really secure environment (CCTV surveillances work ground, 1 metre-of-concrete wall, high secure access control etc ... see common criteria for details). As soon as a TSM would hand the keys to a none trusted party, the chain-of-trust for this secure element is broken. In order not deal with a different key for each phone, there is the possible to rest the keys of the secure element to standards one. From this time on (= after unlocking) the secure element will not be trusted any more. No application provider (eg. MasterCard, Visa, SNCF, ...) will never ever put his application there.

You can use GlobalPlatform Commands set to set new keys for the secure element and start a new chain-of-trust. But anyhow, in this case you have to prove your trustworthiness to other. This means, you will receive some kind of certifications, which are very expensive (probable a six-digit-amount in USD).

Who do I have to talk to in order to load applications into my phone?

Unlock your secure element and load the applications there yourself. That's the most simple way.

As the TSM (trusted service managers) are in charge of the secure element, they are able to do so. Propably, managing one phone is a bit costly, as the TSM usually deal with large number of phones. Trials with around 1000 phones are a suitable size for a TSM (like the one in NYC or Strassbourg/Caen).

How do we develop secure applications in the future, when all phones will have a different set of keys?

Each secure element - no matter where it is located: in the phone, on the SIM card, on an SD card - has an unique Serial Number. Each of the Secure Elements in Nokia 6131 NFC do have its own, chip specific ISD (Issuer Security Domain) keys.

What you need is the ISD key of the secure element. This key allows full access to the SE, and therefore it must be kept secret at all times. This allows e.g. a credit card application to be installed on the secure element so that nobody can gain unauthorized access to it - or worse yet, replace it with their own version.

There will be a service available, which builds a connection between the Secure Element and a back end server. The Server initiated calculates the Chip specific ISD keys and establish a Secure Channel Protocol between the SE and the back end. Through this channel, the server inserts new keys to the SE. The new keys in unlocking the SE are made available for the individual developer.

Once you unlock your SE, the chain of trust is broken, and your SE will NOT BE TRUSTED any more. Not ever. You can no longer install e.g. a real credit card application on it. You can do the unlocking on a virgin phone, but once you've installed a secure application on the SE, the unlocking is not allowed any more.

There will be a service which will allow you to unlock your SE for development work.

If I sign a simple MIDlet, can it talk to for instance a Paypass application?

Essentially yes, if PayPass application is installed in the Secure Element. However, in order to develop something useful, you need access to the Paypass specifications. Secure Element Applets may - by the Applet's own design - allow free access to some of the Applet's services and require authentication to other services.

In the future - with a fully implemented JSR177 - the secure element is able holdinformation on ACP (Access Condition Policy). The ACP defines wich J2ME Midlet is able to communicate with which JavaCard Applet. (see JSR177 Specifications, Appendix A for details)

What are the keys used with the Mifare NDEFTagConnection?

There are essentially 2 keys used for reading NDEF tags:

- keys used for the MAD area, which are: A0A1A2A3A4A5

- keys used for the NDEF area, which are: D3F7D3F7D3F7

Where I can find more information about JavaCard technology?

For more information about Java Card technology, read related literature, for example *Java Card Technology for smart cards: architecture and programmer's Guide* by Zhique Chen. ISBN 0-201-70329-7

Also the Sun Java Forum has a dedicated area for questions and answers on JavaCard

SDK functionality

Is the emulator capable of simulating the internal secure element functionality of the phone?

The internal secure element is a simulated one on the emulator. The implementation of the Nokia 6131 NFC SDK does not contain actual functionality of the real secure element. Also, implementations - by end user - of simulated secure elements are not valid for actual smart-cards.

Are the internal connections limited to a certain domain? Are there any permissions that need to be set in order to use connections to the internal Secure Element?

You don't need to set any permission to access the secure element or to use the ISO14443Connection. However, if you want to access the phone's secure element, the MIDlet has to be in the Trusted 3rd party security domain (has to be signed with a trusted third party certificate).

If the MIDlet is not signed with a third party certificate, you will get a security exception when you are running the MIDlet.

SDK APIs

How can I use the Mifare Standard API?

In order to utilize Mifare Standard API it is mandatory to update the phone software to the latest version, 5.11.

Nokia 6131 NFC software can be updated to version 5.11 in Nokia Service Points. You can find the closest Service Point through Nokia web pages under your local Get support and software pages.

Nokia 6131 NFC software 5.11 will not be available through Nokia Software Updater.

How can I get an ISO14443Connection to the internal secure element of the phone?

Nokia 6131 NFC allows MIDlets to access the internal secure element of the phone. The internal connections are opened by directly calling Connector.open() and using the "internal.se.url" system property. The DiscoveryManager isn't needed.

Here is a sample code for how to open an internal ISO14443 connection, using the javax.microedition.contactless.sc.ISO14443Connection interface:

String uri = System.getProperty("internal.se.url");

ISO14443Connection iseConn = (ISO14443Connection) Connector.open(uri);

How can I get a MiFare Standard connection to the internal secure element of the phone?

Nokia 6131 NFC allows MIDlets to access the MiFare side of the internal secure element of the phone. The internal connections to the MiFare side are opened by directly calling Connector.open() and using the “internal.mf.url” system property. The DiscoveryManager isn't needed.

Here is a sample code for how to open an internal MiFare Standard connection, using the com.nokia.nfc.nxp.mfstd.MFStandardConnection interface:

String uri = System.getProperty("internal.mf.url");

MFStandardConnection mfStdConn = (MFStandardConnection) Connector.open(uri);

Does the Nokia 6131 NFC SDK support NFCIP-1 connection? If so, how can I use that?

The JSR 257 provides an API extension for NFC peer to peer connections. The com.nokia.nfc.p2p package contains the NFCIPConnection interface for communication between two NFCIP devices.

The connection mode (either target or initiator) is decided when opening the connection using the Connector class. An initiator mode connection can be opened with the URL nfc:rf;type=nfcip;mode=initiator and a target mode connection with nfc:rf;type=nfcip;mode=target. Note that the Connector.open(java.lang.String) method call blocks until a NFCIP device is found. Notifications about a NFCIP device cannot be received by registering a TargetListener to DiscoveryManager.

One can define an additional timeout value in the connection URL. The timeout is defined in milliseconds. For example nfc:rf;type=nfcip;mode=initiator;timeout=5000 will wait 5 seconds for a target NFCIP device. Value of 0 means that the open call will block indefinitely. Value 0 is also the default value. In addition to the timeout parameter in the URL string one must also call the Connector.open(String, int, boolean) method and use true as the third parameter to enable timeouts. The second integer parameter is ignored.

In initiator mode one has to first send data, then receive data, send data, receive data, ... One can do as many send-receive sequences as necessary. For every send call there must be a receive call.

In target mode the communications is otherwise similar but one must first receive data and only then send data.

Trying to open another Contactless Communication API related connection will abort a blocking NFCIP Connector.open() call. It is possible to cancel opening a NFCIPConnection by defining a small timeout value to the URL.

If the correct method isn't called an ContactlessException will be thrown.

How can I automatically launch an application when touching an RFID tag?

Nokia 6131 NFC supports the JSR-257 PushRegistry connections as defined in JSR-257 specification 1.0 Appendix B. You can download this from the Java Community Process site – it contains a lot of useful information about PushRegistry (NDEF record push, Secure Element push, possible URLs for NDEF record push, etc.)

Briefly - if a MIDlet is launched by touching an NDEF tag and an NDEFRecordListener is registered to DiscoveryManager using the appropriate NDEF record type within thirty seconds of getting the DiscoveryManager instance, then the listener will be notified and the recordDetected() parameter will contain the NDEF record that triggered the launch.

In order to launch the MIDlet by touching a tag and also get the information on the tag after the MIDlet has been launched, you must implement NDEFRecordListener and its method recordDetected and also, in the constructor, add the NDEFRecordListener to the DiscoveryManager with the corresponding NDEFRecordType.

How can I automatically launch an application when touching another Nokia 6131 NFC?

Similar to launching application by touching tag the PushRegistry can be used to launch application when touching another Nokia 6131 NFC. You need to use "nfc:undefined_format" as connection URL parameter to use this functionality.

Note that when "nfc:undefined_format" is registered application is launched also when tags that doesn't contain NDEF or NTIP data are touched.

Can polling be changed with a midlet?

Not really. An active midlet will make the phone poll even while it is closed, but other than that, power management is done by the

phone itself. Note that it is possible to turn off the polling from the NFC settings menu.

What target types are supported by JSR-257 implemented in Nokia 6131 NFC SDK?

The supported target type values are:

- TargetType.ISO14443_CARD for ISO 14443-4 compliant smart cards accessed using APDU commands

- TargetType.NDEF_TAG for a tag that contains NFC Forum formatted data

- TargetType.RFID_TAG for general RFID tags

PLEASE NOTE: Visual tags are not supported by the Nokia 6131 NFC SDK, nor Nokia 6131 NFC mobile phone.

What are the NDEF RTDs supported by Nokia 6131 NFC?

The following NDEF RTDs are supported by native NFC application and JSR-257 implemented in Nokia 6131 NFC:

```
Type name format Type name Description
-------------------------------------------------------------------------
MIME text/x-vCard Business cards
MIME text/x-vCalendar Calendar notes
NFC Forum RTD urn:nfc:wkt:Sp Smartposters
NFC Forum RTD urn:nfc:wkt:U               URI records
NFC Forum Ext Type urn:nfc:ext:nokia.com:bt        Bluetooth record
                                                  (for printing/image frame)
```

The following NDEF RTDs are reserved for native applications and cannot be used for PushRegistry registrations:

```
Type name format Type name Description
-------------------------------------------------------------------------
MIME text/x-vCard Business cards
MIME text/x-vCalendar Calendar notes
NFC Forum RTD urn:nfc:wkt:Sp Smartposters
```

What is the tag format to connect the Nokia 6131 NFC with a Bluetooth imaging device?

Please note that this is proprietary implementation for Nokia 6131 NFC so it might not work in any other phone!

The NDEF Record type must be EXTERNAL_RTD and the type name "urn:nfc:ext:nokia.com:bt"

For the payload you need to make following byte array:

```
Name Size Description
-------------------------------------------------------------------------
Configuration type 1 byte 0x00 = Discovery only
                                 0x01 = PIN
                                 0x02 = Public key

Buetooth address 6 bytes Bluetooth address.

Class of Device (CoD)  3 bytes Only imaging tags are supported by Nokia 6131
                                 NFC, so bytes should in following bit format:
```

```
                                    xxxxxxxx xxx00110 1xxxxxxx. For example 0x00
                                    0x06 0x80

Authentication info 16 bytes Depends on the configuration type.

Short name length 1 byte Length of short name in bytes.

Short name n bytes Text displayed on screen upon discovery.
```

There is also an example code about this: Write and read Bluetooth NDEF tag

SDK Known Issues

- Release notes:

Nokia 6131 NFC SDK Known Issues:

- DESFire does not work properly on emulator/external readers.

Nokia 6212 NFC SDK Known Issues:

- NFC Manager does not work on WIndows 7. This discussion provides a solution to that problem.

Java Code Examples for Nokia 6131 NFC

Where I can find Java examples for Nokia 6131 NFC?

The Nokia 6131 NFC SDK contains example Java applications. Additionally more examples can be found on Portal:Java ME NFC Articles

WIMA conference in Monaco, 2007

Please find below the material presented in WIMA conference by Mikko Saarisalo, Nokia. (In absence of better place for these files)

- Java examples: File:Examples.zip
- Presentation: WIMA 2007 04 Mikko Saarisalo.pdf

All questions and answers from WIMA conference has been moved to other sections on this page.

Join to connect event in Finland, 2007

Questions & Answers from Join to connect event in Finland, 2007

Nokia NFC Unlock Service MIDlet

Use the Nokia NFC Unlock Service MIDlet to unlock the secure element on the Nokia 6131 NFC.

- Read Me
- Download