**NFC и беспроводные интерфейсы приложений**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | |  | | --- | |  | |   Технология NFC (Near-field Communication) предоставляет коротко диапазонные коммуникации с различными устройствами, в частности с мобильными. NFC может использоваться в качестве платежного средства.  NFC представляет собой стандарт, принятый NFC-форумом (2004 год), всемирным консорциумом разработчиков оборудования, программ/приложений, компаний - эмитентов кредитных карт, сервис-провайдеров и банковских работников (всего более 80 компаний и организаций). Смотри <a href=http://ru.wikipedia.org/wiki/Near\_Field\_Communication>Near Field Communication</a>),  а также <a href=http://java.sun.com/developer/technicalArticles/javame/nfc/>An Introduction to Near-Field Communication and the Бесконтактные коммуникации API, [*C. Enrique Ortiz*](http://java.sun.com/features/authors.html#ortiz)*, June 2008*</a>.  **NFC** — простое расширение стандарта ISO 14443, которое объединяет интерфейс и считыватель в одно устройство. NFC может поддерживать связь с существующими смарт-картами, считывающими устройствами стандарта ISO 14443, и другими устройствами NFC. Эта техника совместима с существующей инфраструктурой бесконтактных карт, уже использующейся в общественном транспорте и платежных системах.  NFC работает на частоте 13.56 MHz (диапазон ISM (*Industrial, Scientific and Medical radio Bands* — Промышленные, научные и медицинские частоты) обеспечивает скорость передачи до 424 кбит/c. NFC-коммуникации запускаются, когда NFC-совместимые устройства оказываются на расстоянии около 4 см друг от друга. Так как расстояние минимально, утечка данных практически исключена. В NFC связь поддерживается за счет взаимной индукции рамочных антенн.  Радио интерфейс для NFC стандартизирован в:   * ISO/IEC 18092 / ECMA-340 : Near Field Communication Interface and Protocol-1 (NFCIP-1) * ISO/IEC 21481 / ECMA-352 : Near Field Communication Interface and Protocol-2 (NFCIP-2)   Максимальн6ое рабочее расстояние для стандартных антенн не превышает 20 см, а скорость передачи данных может принимать значения 106, 212 или 424 кбит/сек.  Ниже в таблице проведено сравнение характеристик известных короткодиапазонных технологий:   |  | | --- | | http://java.sun.com/developer/technicalArticles/javame/nfc/figure1.gif  Рис. 1. Сравнение NFC с другими корокодиапазонными коммуникационными технологиями (источник: NFC Forum) |    Сравнение NFC с другими технологиями показывает, что это самая короткодиапазонная техника, ориентированная на человека. Некоторые другие технологии имеют сходные параметры, например, RFID, но другие скорее являются дополняющими по отношению к NFC; к таким технологиям можно отнести Bluetooth и Infrared.   |  | | --- | | **NFC стандарты** Стандарты **ISO 14443** тип A и тип B + **FeliCa** представляют собой четыре части международного стандарта для бесконтактных платежных карт. Стандарт **ISO 18092** определяет режимы работы интерфейса NFC и протокол. |   **Режимы NFC**  NFC-форум определил три коммуникационных режима, как это показано ниже:   |  | | --- | | http://java.sun.com/developer/technicalArticles/javame/nfc/figure2.gif  Рис. 2. Режимы работы NFC-коммуникаций (источник: NFC Forum) |     где:   * Режим **Peer-to-Peer** определен для коммуникаций устройство-устройство. Заметим, что этот режим не поддерживается API бесконтактных коммуникаций. * Режим **чтения/записи** позволяет приложениям обмениватьсяNFC-сообщениями. Заметьте, что этот режим не является безопасным. Этот режим поддерживается API бесконтактные коммуникации. * Режим эмуляцииNFC-картыпозволяет NFC-телефонной гарнитуре работать в качестве стандартной смарт-карты. Этот режим безопасен и поддерживается API бесконтактных коммуникаций.   **API бесконтактных коммуникаций**  Java-спецификация API бесконтактных коммуникаций (Nokia) определена документами Java-сообщества JSR-257, которые фиксируют базовые функции бесконтактных коммуникаций. API состоит из пяти Java-пакетов:  **Таблица 1 - Java-пакеты JSR 257**   |  |  |  |  | | --- | --- | --- | --- | | **Java-пакет** | **Интерфейсы** | **Классы** | **Исключения** | |  | | | | | javax.microedition.contactless Обязательная часть пакета, которая содержит все общие функции и классы | TagConnection TargetListener TargetProperties TransactionListener | DiscoveryManager TargetType | ContactlessException | | javax.microedition.contactless.ndef Опциональный пакет для коммуникаций с форматами NDEF | NDEFRecordListener NDEFTagConnection | NDEFMessage NDEFRecord NDEFRecordType |  | | javax.microedition.contactless.rf Опциональный пакет для коммуникаций сRFID (а не с форматами NDEF) | PlainTagConnection |  |  | | javax.microedition.contactless.sc Опциональный пакет для коммуникаций с внешними смарт-картами | ISO14443Connection |  |  | | javax.microedition.contactless.visual Опциональный пакет для чтения и генерации визуальных меток | ImageProperties VisualTagConnection | SymbologyManager | VisualTagCodingException | |  | | | |     \* Заметим, что обязательным пакетом является только javax.microedition.contactless  API бесконтактных коммуникаций позволяет вам осуществлять выявление и обмен данными с бесконтактными объектами, такими как метки NDEF, RFID и внешние смарт-карты. API осуществляет также поддержку визуальных меток. На диаграмме ниже проиллюстрированы взаимоотношения между различными классами и интерфейсами API:   |  | | --- | | http://java.sun.com/developer/technicalArticles/javame/nfc/figure3.gif  Рис. 3: Взаимодействие API бесконтактных коммуникаций (источник: спецификация JSR-257) |     **Анатомия бесконтактных коммуникаций API MIDlet**  Рассмотрим элементы типового мобильного Java-приложения (MIDlet), которое использует принципы API беспроводных коммуникаций. Ниже представлена иллюстрация этого:   |  | | --- | | http://java.sun.com/developer/technicalArticles/javame/nfc/figure4.gif  Рис. 4. Анатомия API бесконтактных коммуникаций, базирующихся на Java MIDlet (Источник: CEnriqueOrtiz.com) |    Где мы имеем следующие типовые элементы:   * Java реального времени с реализацией JSR-257, * Приложение MIDlet, работающее в телефонной гарнитуре, * Транспондер RFID/NFC, контроллеры и среда передачи, * SIM-карта, а также элементы безопасности.  |  |  | | --- | --- | | **Внешние читающие средства** | **Элементы безопасности** | |  | | | В число внешних читающих устройств входят бесконтактные платежные считыватели торговых автоматов, билетные системы на транспорте, визуальные метки, такие как NFC, RFID и штриховые коды или смарт-карты | Элементы безопасности (<b>SE</b>) могут быть внутренними или внешними; примером элемента безопасности является смарт-карта, базирующаяся на Java. MIDlets может воспринимать элементы безопасности посредством служб безопасности и доверия API (SATSA), и/или бесконтактных коммуникаций API (JSR 257). Внешние системы считывания осуществляют доступ к внутренним элементам безопасности непосредственно через схему RFID (используя режим эмуляции карты). | |  | |     **Использование API бесконтактных коммуникаций**  API бесконтактных коммуникаций позволяет вам you выявлять и обмениваться данными через поддерживаемые бесконтактное радио и визуальные метки. Приложения, использующие API бесконтактных коммуникаций обычно следуют схеме, представленной ниже:   |  | | --- | | http://java.sun.com/developer/technicalArticles/javame/nfc/figure5.gif  Рис. 5. Типовая блок-схема реализации приложения бесконтактных коммуникаций |    где:   1. На первом этапе для приложения выясняется, какие типы объектов поддерживаются телефонной гарнитурой, 2. Для каждого поддерживаемого типа приложение может регистрировать прослушивающее устройство, чтобы получить уведомление об активности. В качестве альтернативы, приложение может с помощью *PushRegistry* зарегистрировать активность объекта; поддерживаемыми активностями объекта является NDEF и элемент безопасности в режиме эмуляции карты, 3. При приближении объектов они детектируются (выявляются), после чего активируются соответствующие считывающие средства. Альтернативой является активация MIDlet со стороны PushRegistry, 4. Для каждого выявленного объекта приложение может выяснить его свойства, 5. Приложение может установить соединение, читать, записывать и осуществлять информационный обмен с выявленным объектом, 6. По завершении, с целью освобождения ресурсов, приложение закрывает любые установленные ранее соединения.   **API бесконтактных коммуникаций и обобщенная схема соединения**  API бесконтактных коммуникаций следующим образом расширяет общую схему соединения (GCF):   |  | | --- | | http://java.sun.com/developer/technicalArticles/javame/nfc/figure6.gif  Рис. 6. API бесконтактных коммуникаций и общая схема соединения |    API бесконтактных коммуникаций определяет следующие интерфейсы:   * TagConnection определяет базовый интерфейс для всех RFID, смарт-карт и NFC-соединений:   + ISO14443Connection расширяет TagConnection и используется для бесконтактного соединения с ISO 14443-4 совместимыми бесконтактными смарт-картами.   + PlainTagConnection расширяет TagConnection и используется для бесконтактного соединения с (не-NFC) метками.   + NDEFTagConnection расширяет TagConnection и используется для бесконтактного соединения с RFID-метками, имеющими формат, заданный NFC-форумом, и с смарт-картами * VisualConnection используется для бесконтактного соединения с визуальными метками, такими как штриховой код   Дополнительную информацию можно получить в статье <a href= http://developers.sun.com/mobility/midp/articles/genericframework/>The Generic Connection Framework</a>.  **Выявление и прослушивание поддерживаемых объектов**  Для выявления объектов, которые поддерживаются платформой, использовавшей метод DiscoveryManager.getSupportedTargetTypes(), который возвращает массив TargetTypes. Затем для каждого типа объекта регистрируется прослушивающее устройство объекта. Следующий фрагмент кода демонстрирует метод registerTargetListener(), который запрашивает платформу о поддерживаемых объектах, затем об объектах типа ISO 14443-4 (совместимых с бесконтактными смарт-картами), он регистрирует устройство прослушивания объекта:  **Листинг 1 – Выявление объектов и их регистрация**   |  | | --- | | import javax.microedition.contactless.TargetListener;  :  DiscoveryManager dm = DiscoveryManager.getInstance();  :  /\*\*  \* Discover supported targets, registers listeners  \*  \* @param targetListener the target listener  \*/  public void registerTargetListeners(TargetListener targetListener) {  // Discover supported types  TargetType[] tp = DiscoveryManager.getSupportedTargetTypes();  try {  // Register listener for each of the supported types  for (int i=0; i<tp.length; i++) {  if (tp[i].equals((TargetType.ISO14443\_CARD))) {  dm.addTargetListener(  targetListener, TargetType.ISO14443\_CARD);  } else...  :  :  }  }  } catch (Exception e) {  // ...  }  } |    Заметим, что API бесконтактных коммуникаций определяют следующие типы объектов: ISO14443\_CARD, NDEF\_TAG, RFID\_TAG и VISUAL\_TAG.  Когда нужные объекты выявлены и соответствующие приемные устройства зарегистрированы, как это показано выше, программная реализация будет использовать метод TargetListener.targetDetected(TargetProperties[]), если детектирован новый объект:  **Листинг 2 – Прослушивание приемных устройств зарегистрированных объектов**   |  | | --- | |  | | import javax.microedition.contactless.TargetListener;  :  /\*\*  \* A new target has been detected. This method is invoked by  \* the platform.  \*  \* @param prop the properties for the detected target  \*/  public void targetDetected(TargetProperties[] prop) {  for (int i = 0; i < prop.length; i++) {  // Get UID  String uid = prop[i].getUid();  // Get Connection Classes  Class[] classes = prop[i].getConnectionNames();  // Get Target Types  TargetType[] types = prop[i].getTargetTypes();  // Connect to each Target  String url = prop[i].getUrl();  try {  // Open NDEFTagConnection to the target  NDEFTagConnection conn =  (NDEFTagConnection) Connector.open(url);  :  :  } catch (IOException e) {  // ...  }  }  } |    Метод targetDetected() обычно выполняет следующее:   * Метод targetDetected получает TargetProperties для детектированных объектов, * Для каждого детектированного объекта, его URL извлекается из перечня свойств объекта, * Соединение с объектом осуществляется с помощью GCF, * Производится обмен данными, * Входные сообщения обрабатываются в соответствии с их атрибутами, * По завершении, все ресурсы освобождаются, а соединения разрываются.   Следует заметить, что чтение меток RFID требует понимания того, как работают команды ISO14443 I/O и ADPU.  **Прослушивание специфических NDEF-объектов**  API бесконтактных коммуникаций обеспечивает также распознавание специфических NDEF-объектов без детального знания каких-либо подробностей; единственно, что нужно знать, это тип записи (сообщения) и процедуру их обработки, это существенно упрощает логику коммуникаций информационного обмена. Чтобы прослушивать NDEF-объекты, используется интерфейс NDEFRecordListener и его метод recordDetected(NDEFMessage ndefMessage). Регистрация слушателя NDEF-записей осуществляется с привлечением метода addNDEFRecordListener(listener, recordType). Следующий фрагмент кода иллюстрирует это:  **Листинг 3 – Листинг для объектов NDEF**   |  | | --- | |  | | import javax.microedition.contactless.ndef.NDEFRecordListener;  :  DiscoveryManager dm = DiscoveryManager.getInstance();  :  // Register NDEF\_TAG target (smart poster) to discover  try {  NDEFRecordType rt = new NDEFRecordType(  NDEFRecordType.NFC\_FORUM\_RTD, "urn:nfc:wkt:Sp");  dm.addNDEFRecordListener(this, rt);  } catch (IllegalStateException e) {  :  } catch (Exception e) {  :  } |   Ниже определены типы NDEF-записей:   * **EMPTY**- идентификатор записи формата типа имени для пустого рекорда. * **EXTERNAL\_RTD** – идентификатор записи формата типа имени для имен типов специальных рекордов приложения, которые следуют спецификациям NFC-форума. * **MIME** - Record type name format identifier for the MIME type defined in the RFC 2046. * **NFC\_FORUM\_RTD** - идентификатор записи формата типа имени для описаний типов рекордов NFC-форума. * **UNKNOWN** - идентификатор записи формата типа имени для неизвестных типов рекордов. * **URI** - идентификатор записи формата типа имени для типов URI, определенных в RFC 3986.  |  | | --- | | Из спецификации JSR 257. Если формат имеет тип EMPTY или UNKNOWN имя должно иметь значение нуль. Типы имен записей NFC\_FORUM\_RTD и EXTERNAL\_RTD должны следовать правилам, определенным в RTD- спецификации NFC-форума. Имя типа рекорда должно содержать только символы из набора US\_ASCII. |   **Обработка NDEFMessages**  Если NDEF-приемник зарегистрирован, платформа воспользуется методом recordDetected(NDEFMessage ndefMessage) для запрошенного NDEF, передавая в качестве аргумента сообщение NDEF для объекта NDEF, который становится активным и видимым. Метод recordDetected() получает запись, тип записи и другую информацию из NDEF-сообщения, и обрабатывает сообщение согласно его атрибутам.  **Листинг 4 – Обработка сообщений NDEF-объектов**   |  | | --- | |  | | /\*\*  \* Вызывается платформой, когда устройство бесконтактного объекта выявляет  \* запрошенный тип рекорда NDEF.  \*  \* @param ndefMessage the NDEF message to process  \*/  public void recordDetected(NDEFMessage ndefMessage) {  // Get records and record types from NDEF Message  NDEFRecordType[] rTypes = ndefMessage.getRecordTypes();  NDEFRecord[] records = ndefMessage.getRecords();  for (int i=0; i<records.length; i++) {  // Handle data, based on type of NDEFMessage  NDEFRecordType t = recordTypes[i];  NDEFRecord r = records[i];  byte[] id = r.getId();  long len = r.getPayloadLength();  byte[] p = r.getPayload();  // Process the record  // ...  }  } |    Обработка тела сообщения будет зависеть от вашего приложения; например, после чтения URL, приложение может послать некоторую информацию этому URL, или возможно откроет этот URL на web-браузере.  **Использование уведомлений об активности эмуляции карты**  В режиме эмуляции карты, элемент безопасности прибора взаимодействует с внешним устройством чтения RFID. Приложения уведомляются о таких транзакциях, но не участвуют в них. Это проиллюстрировано ниже:   |  | | --- | | http://java.sun.com/developer/technicalArticles/javame/nfc/figure7.gif  Рис. 7. Уведомления активности эмуляции карты |    где:   1. Внутренний элемент безопасности взаимодействует с внешним устройством чтения, 2. Приложение (MIDlet) уведомляется, когда детектировано внешнее устройство чтения, 3. Если необходимо, приложение взаимодействует с элементом безопасности, используя API бесконтактных коммуникаций интерфейса ISO14443, или SATSA, если такое имеется.   Обмены между элементом безопасности и внешним считывателем открыты для приложения, и происходят с использованием APDU-команд, точно также как Java-карты взаимодействуют с внешними читающими устройствами. Более подробную информацию можно найти в “[An Introduction to Java Card Technology - Part 1](http://java.sun.com/javacard/reference/techart/javacard1/)” (ниже в разделе «Источники”). Заметим, что коммуникации между приложением и безопасным элементом требуют знания специальных безопасных апплетов, используемых в безопасном элементе.   |  | | --- | | APDU-команды определены в спецификации ISO7816-4. |   На следующей диаграмме, взятой из статьи “[An Introduction to Java Card Technology - Part 1](http://java.sun.com/javacard/reference/techart/javacard1/)”, иллюстрирует типовое взаимодействие приложения Java-карты (в этом случае MIDlet играет роль считывателя), и элементов безопасности (играющих роль "стороны карты"):   |  | | --- | | http://java.sun.com/developer/technicalArticles/javame/nfc/figure8.gif  Рис. 8. Типовые элементы приложения для Java-карты. Прикладные сценарии бесконтактных коммуникаций (Источник: CEnriqueOrtiz.com) |    где:   * В бесконтактной NFC-гарнитуре, левая сторона может быть внутренним считывателем, сам MIDlet, или и внешним читающим устройством (посредством режима эмуляции NFC-карты), * Правая сторона, "карта", может быть внутренним или внешним элементом безопасности, который доступен через SATSA или JSR-257, или через оборудование RFID, * Все обмены осуществляются с помощью APDU-команд.   **Регистрация и обработка активности, сопряженной с эмуляцией карты**  Как было отмечено выше, в случае режима эмуляции карты, приложения могут уведомляться о том, что детектировано внешний считыватель, но само приложение не участвует в транзакциях. Кроме того, если необходимо, приложение осуществляет обмены с элементом безопасности, используя команды APDU, и для этой цели приложение должно знать об апплетах, работающих в элементе безопасности. Чтобы регистрировать уведомления активности эмуляции карты используются интерфейс javax.microedition.contactless.TransactionListener и его метод externalReaderDetected(byte slot). Регистрация транзакции прослушивателя выполняется путем вызова метода dm.addTransactionListener(...). Ниже приведенный фрагмент кода иллюстрирует это:  **Листинг 5 – Регистрация и обработка активности, сопряженной с эмуляцией карты**  Для получения более детальной информации о Java-карте и SATSA, смотри раздел Ресурсы в конце данной статьи.  **Использование PushRegistry для запуска NFC-приложений**  Automatic application activation or startup is a very important aspect for the success and acceptance of NFC-based applications, as it allows for a better user experience; by just coming into proximity of (or touching) a reader or another NFC device, the application is automatically started. The PushRegistry facility that is part of the Mobile Information Device Profile (MIDP) provides for the automatic application-launch based on timers and connection activity. Заметим, что API бесконтактных коммуникаций расширяет PushRegistry для запуска приложения только для 1) типов рекордов NDEF и 2) для элементов безопасности (активности) коммуникаций режима эмуляции карты.  URL соединений, используемые для регистрации должны следовать соответствующим требованиям, как это определено в спецификации:   * URL format for NDEF push connections: "ndef:"<record\_type\_format>?name=<record\_type\_string>   Где:   * + <record\_type\_format> имеет вид "rtd", "external\_rtd", "mime", или "uri",   + <record\_type\_string> имеет вид UTF-8 строки, которая представляет собой имя типа записи, как это специфицировано конкретной реализацией, например, urn:nfc:wkt:Sp (Nokia Smart-posters) * Формат URL для эмуляции карты имеет вид: "secure-element:"?aid=<aid\_string> Где:   + <aid\_string> является строкой, содержащей идентификатор апплета (цифровые символы), в соответствии со спецификацией ISO7816-5.   Чтобы гарантировать получение уведомлений при запуске приложения в случае выявления NDEF-записей или активности эмуляции карты, нужно зарегистрировать NDEFRecordListener и TransactionListener сразу как только они начали работать.  **Соображения безопасности**  API бесконтактных коммуникаций являются объектом политики безопасности базовой мобильной платформы. Для MIDP, следующие методы или операции требуют разрешения:  **Таблица 2 - API бесконтактных коммуникаций MIDP Security Permissions**   |  |  | | --- | --- | | **Метод или операция** | **Разрешение** | |  | | | DiscoveryManager.getInstance() | javax.microedition.contactless.DiscoveryManager | | Write NDEFMessage message | javax.microedition.contactless.ndef.NDEFTagConnection.write | | Opening NDEFTagConnection | javax.microedition.io.Connector.ndef | | Opening PlainTagConnection | javax.microedition.io.Connector.rf | | Opening ISO14443Connection | javax.microedition.io.Connector.sc | | Opening VisualTagConnection | javax.microedition.io.Connector.vtag | |  | |     В этой модели, вы запрашиваете разрешения через JAD-файл или JAR-манифест, путем формирования MIDlet-Permissions, используя имена разрешений из выше приведенной таблицы. Попытки использовать операции с ограничениями, без соответствующего разрешения вызовет блокировку системы через SecurityException.  Для практических реализаций API бесконтактных коммуникаций на основе CDC-профайлов, приложение должно использовать проверки безопасности, базирующиеся на java.security.Permission. Следующие методы требуют разрешения:   * DiscoveryManager.getInstance() * NDEFTagConnection.write(NDEFMessage message)   Другие применения NFC в будущем могут включать:   * Электронная покупка билетов на самолет, поезд или на концерт * Удостоверения личности * Электронные деньги * Электронные ключи — ключи от машины, дома, гостиничного номера, и т. д.   **Источники**   * <a href=http://jcp.org/en/jsr/detail?id=257>[JSR 257: Бесконтактные коммуникации API](http://jcp.org/en/jsr/detail?id=257)</a> * <a href=http://jcp.org/en/jsr/detail?id=177>[JSR 177: Security and Trust Services API for J2ME](http://jcp.org/en/jsr/detail?id=177)</a> * <a href= http://www.nfc-forum.org/home>[The NFC Forum](http://www.nfc-forum.org/home)</a> * <a href=http://www.oracle.com/technetwork/java/javacard/documentation/index.html>[An Introduction to Java Card Technology - Part 1](http://java.sun.com/javacard/reference/techart/javacard1/)</a> * <a href=http://www.oracle.com/technetwork/java/javacard/documentation/index.html>[An Introduction to Java Card Technology - Part 2, The Java Card Applet](http://java.sun.com/javacard/reference/techart/javacard2/)</a> * <a href= http://www.oracle.com/technetwork/java/javacard/documentation/index.html>[An Introduction to Java Card Technology - Part 3, The Smart Card Host Application](http://java.sun.com/javacard/reference/techart/javacard3/)</a> * <a href= http://developers.sun.com/mobility/apis/articles/satsa1/>[The Security and Trust Services API for J2ME, Part 1](http://developers.sun.com/mobility/apis/articles/satsa1/)</a> * <a href= http://developers.sun.com/mobility/midp/articles/pushreg/>[The MIDP 2.0 Push Registry](http://developers.sun.com/mobility/midp/articles/pushreg/)</a> * <a href= http://en.wikipedia.org/wiki/Near\_Field\_Communication>[Near Field Communication (Wikipedia)](http://en.wikipedia.org/wiki/Near_Field_Communication)</a> * <a href= http://www.forum.nokia.com/Develop/>[Forum Nokia - Near Field Communication](http://www.forum.nokia.com/main/resources/technologies/nfc/)</a> |