Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Desarrollo de una guía para la implementación de aplicaciones basadas en NFC

Proyecto Final de Carrera

Ingeniero en Informática

Valencia 28 de Noviembre del 2016

Autor: Antonio Albiñana Martínez
Director: Joaquín Gracia Morán

pagina en blanco
intencionada

Índice

Prólogo

En los últimos años, el desarrollo tecnológico ha propiciado la aparición de nuevas formas de comunicación entre dispositivos.

En este marco se encuadra la tecnología NFC (*Near Field Comunication*), que en España es más conocida como "*contactless*" (sin contacto). Esta tecnología hace referencia a un protocolo/tecnología de comunicación de corto alcance. Es relativamente moderna ya que el primer dispositivo con NFC, el Nokia 6131, salió al mercado en Abril del 2006.

El uso de NFC ha ido avanzando de manera especialmente lenta, sobre todo si se compara con otras tecnologías, como puede ser el pago con tarjetas de crédito o el uso del los teléfonos inteligentes (*smartphones*) como parte esencial de nuestro día a día.

De hecho el incremento en la utilización de NFC se basa principalmente en tres usos:

- Pago de billetes en trasporte público.
- Pago a través de tarjetas de crédito equipadas con esta tecnología.
- Los *smartphones*, bien de forma nativa o incluida en la sim[1] (p. ej. Vodafone).

Pero la tecnología NFC puede usarse en multitud de ámbitos, tanto social como industrial. La publicidad, el control de accesos, la gestión de mercancías o inventario son algunos ejemplos de aplicación.

El objetivo de este proyecto es analizar y presentar esta tecnología y sus características, describiendo sus particularidades y así presentar una aproximación a su uso en distintas plataformas de desarrollo como son Android, Python, Java, Arduino y Raspberry Pi. Además, se han incluido algunos ejemplos de uso.

1 *subscriber identity module (módulo de identificación de abonado), incluye nº de teléfono mas espacio para datos*

Resumen

La idea general del presente proyecto consiste en un acercamiento a las características y especificaciones de NFC y la implementación de su uso con ejemplos de dicha tecnología sobre diferentes plataformas como son Python, Arduino, Raspberry Pi, Android y Java.

Se explica en qué consiste dicha tecnología, cuáles son sus particularidades, sus modos de funcionamiento y como las etiquetas NFC pueden ser utilizadas para realizar diversas acciones de manera automática.

Como NFC se basa en la tecnología RFID, su funcionamiento básico es similar a esta tecnología, aunque NFC además introduce comunicaciones punto a punto y emulación de tarjeta.

De esta manera tenemos los distintos modos de comunicación, que son:

- Punto a punto: Para intercambio de datos en pequeñas cantidades.
- Lectura-escritura. Tiene la capacidad de leer o escribir etiquetas.
- Emulación de tarjeta. Esta configuración utiliza las características del elemento de seguridad incorporado en ta tarjeta NFC como medio de pago.

En este trabajo se describen también los distintos tipos de registros:

- NDEF: Para intercambio de datos.
- URI: Contiene un dirección web.
- Texto: Contiene texto plano.
- Smart-Poster: Puede contener metadatos para realizar acciones.

También se presenta un elemento esencial en la tecnología NFC, como son los los distintos tipos de etiquetas, desde el Tipo 1, hasta el último recién incorporado, denominado NFC-V (Tipo5) y que permite ampliar la funcionalidad de las etiquetas RFID mas utilizadas (ISO-15693).

Otro aspecto importante son los mecanismos de seguridad para gestionar los permisos de lectura/escritura de las etiquetas, y cómo funciona el servicio NFC-SEC

para especificar los mecanismos criptográficos en las comunicaciones NFC seguras.

¿Qué es NFC?

NFC es el acrónimo de "*Near Field Communication*" y hace referencia a un protocolo/tecnología de comunicación de corto alcance, de unos 10-15 cm como máximo.

Funciona en la frecuencia de 13,56 Mhz, y está bajo el estándar RFID. Esta es una frecuencia libre y por lo tanto está exenta de licencias, por lo que su uso no tiene ninguna restricción legal.

NFC puede trabajar sobre el estándar ISO-14443 (RFID), con NFCIP[2]-1 (ISO-18092) y NFCIP-2 (ISO 21481).

Podemos reconocer dos logotipos oficiales para dispositivos NFC.

Además de alguno específico de alguna marca como .

La organización encargada de regular y determinar los estándares de NFC nace en 2004 cuando se crea "NFC Forum"[3] (por Nokia, Philips y Sony), pero no es hasta 2006 cuando sale a la luz la primera especificación técnica de dicha tecnología.

Miembros de este consorcio son en la actualidad Google, Nokia (uno de sus principales impulsores), Visa, Dell, Intel, Microsoft, Samsung, Sony, At&t, PayPal y Mastercad por citar algunos de ellos.

A la vista de las empresas implicadas se ve claramente que uno de los principales objetivos de NFC es el comercio electrónico, aunque no es el único.

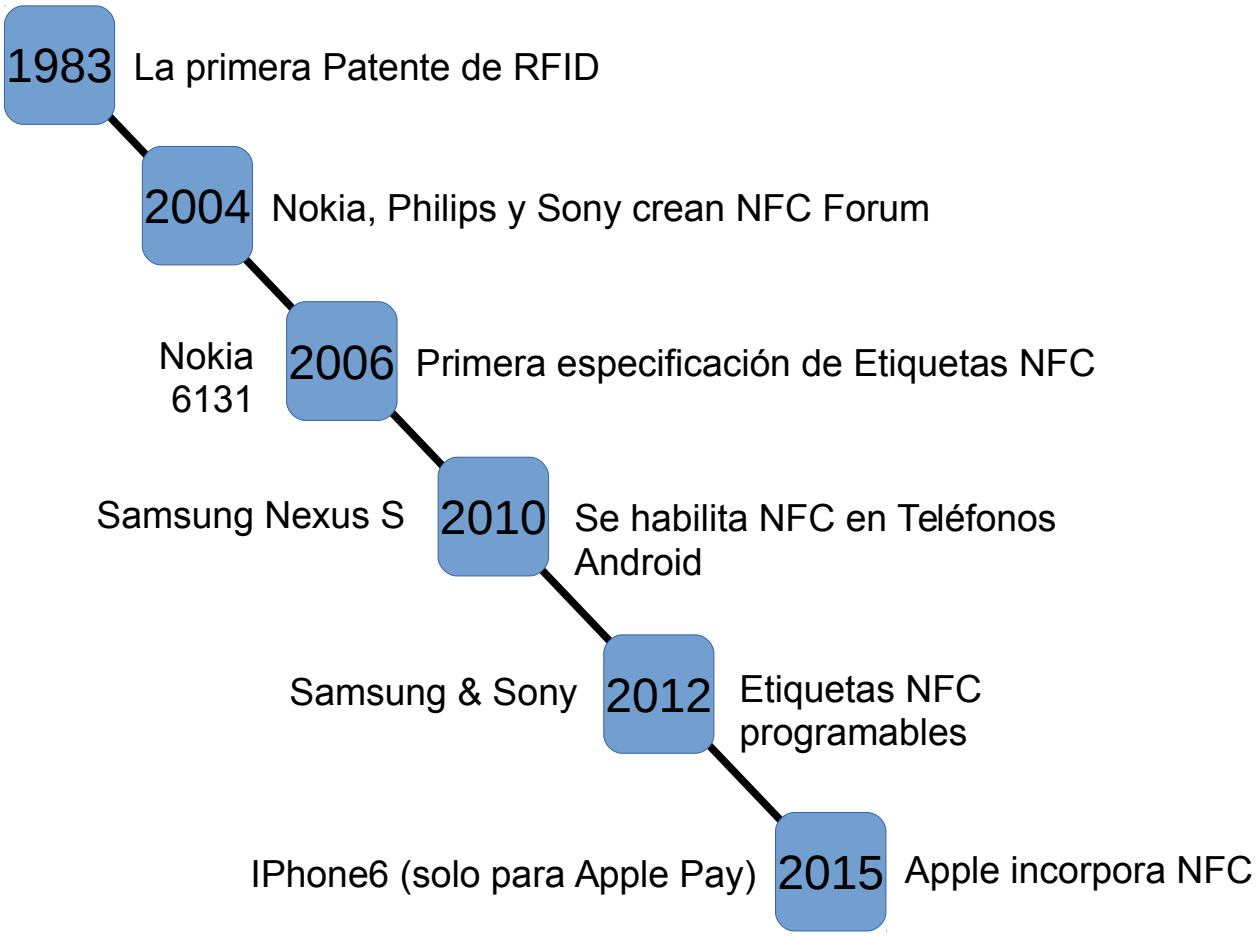
2 *NFC Interface and Protocol.*
3 *http://www.nfc-forum.org/home/*

"NFC Forum", además de la tecnología, define sus objetivos, que son:

- Desarrollar las especificaciones y estándares, así como las arquitecturas y los parámetros de inalterabilidad para el protocolo y dispositivos NFC.
- Fomentar el desarrollo de productos con las especificaciones "NFC Forum".
- Asegurarse de que los productos que dicen llevar NFC cumplen con las especificaciones determinadas por "NFC Forum".
- Educar a los consumidores y empresas a comprender qué es NFC.

La siguiente figura muestra un diagrama temporal de los principales acontecimientos relacionados con esta tecnología[4].

4 *http://nearfieldcommunication.org/history-nfc.html*



Como NFC se basa en los estándares RFID, realizaremos primero un pequeño repaso para ver cómo funciona esta tecnología base, para después describir en detalle el funcionamiento de NFC.

RFID

El modo de funcionamiento de los sistemas RFID (Identificación por Radio Frecuencia) es simple.

RFID define etiquetas, receptores y un sistema de procesamiento de datos.

La etiqueta RFID contiene los datos de identificación del objeto al que se encuentra adherido. Esta etiqueta genera una señal de radiofrecuencia con dichos datos. Existen dos tipos de etiqueta: pasiva o activa. Una etiqueta pasiva recibe la energía del lector y por lo tanto no necesita baterías para su funcionamiento. Una etiqueta activa genera su propia energía.

La señal generada por la etiqueta puede ser captada por un lector RFID, que se denomina receptor. Éste se encarga de leer la información y pasarla en formato digital a la aplicación específica (el sistema de procesamiento de datos) que utiliza RFID.

De esta manera, un sistema RFID consta de los siguientes tres componentes:

- Etiqueta: compuesta por una antena, un transductor radio y un material encapsulado o chip. Existen varios tipos de etiquetas. El chip posee una memoria interna con una capacidad que depende del modelo y varía de una decena a millares de bytes. Existen también varios tipos de memoria:

  Solo lectura: el código de identificación que contiene es único y se personaliza durante la fabricación de la etiqueta.

  De lectura y escritura: la información de identificación puede ser modificada por el lector.

  Anticolisión. Se trata de etiquetas especiales que permiten que un lector identifique varias al mismo tiempo (habitualmente las etiquetas deben entrar una a una en la zona de cobertura del lector).

- Lector de RFID: compuesto por una antena, un transductor y un decodificador. El lector envía periódicamente señales para ver si hay alguna etiqueta en sus inmediaciones. Cuando capta una señal de una etiqueta (la cual contiene la información de identificación de ésta), extrae

la información y se la pasa al subsistema de procesamiento de datos.

– Subsistema de procesamiento de datos: Se encarga de procesar la información recibida de las etiquetas.

Una etiqueta RFID tiene este aspecto:

Pero puede encapsularse dentro de contenedores que le permiten poder funcionar en entornos hostiles donde otros sistemas encontrarían problemas.

Los estándares que regulan RFID son:

- ISO-14443 nombra las diferentes partes y componentes.
- ISO-14443-1 define el tamaño y las características físicas;
- ISO-14443-2 define las características de los campos para la comunicación bidireccional.
- ISO-14443-3 define la fase de inicialización de la comunicación y los protocolos de anticolisión.
- ISO-14443-4 especifica el protocolo de transmisión.

Funcionamiento básico de NFC

Tal y como se ha comentado, esta tecnología parece destinada a desbancar a otras que están funcionando en la actualidad, como los códigos QR (*Quick Response Code*) y similares.

Actualmente, cuando se quiere leer un código QR, el usuario tiene que:

1. Activar el teléfono (seguramente estará con la pantalla apagada).
2. Buscar una aplicación determinada.
3. Abrir dicha aplicación.
4. Apuntar con la cámara al dibujo que representa el código QR.
5. Esperar a que la aplicación interprete correctamente la información.

Con NFC los pasos que hay que seguir son:

1. Acercar el teléfono y automáticamente se empezarán a realizar las acciones determinadas en la etiqueta que se acaba de leer.

   Estas acciones pueden ser desde abrir el navegador para ir a una página web, hasta cambiar configuraciones internas como puede ser poner en silencio o encender/apagar el bluetooh porque entramos o salimos del coche (Sony Xperia[5] proporciona esta opción).

Como puede observarse, la sencillez, funcionalidad y versatilidad de NFC es bastante importante.

Básicamente, la tecnología NFC presenta dos modos de funcionamiento:

- Pasivo, en las que solo un dispositivo genera el campo electromagnético, y el otro se aprovecha de la modulación para poder trasferir los datos. En este caso, el dispositivo que inicia la comunicación es el que genera dicho campo.
- Activo, en el que ambos dispositivos generan campos electromagnéticos. En este caso, ambos dispositivos necesitan una fuente de energía.

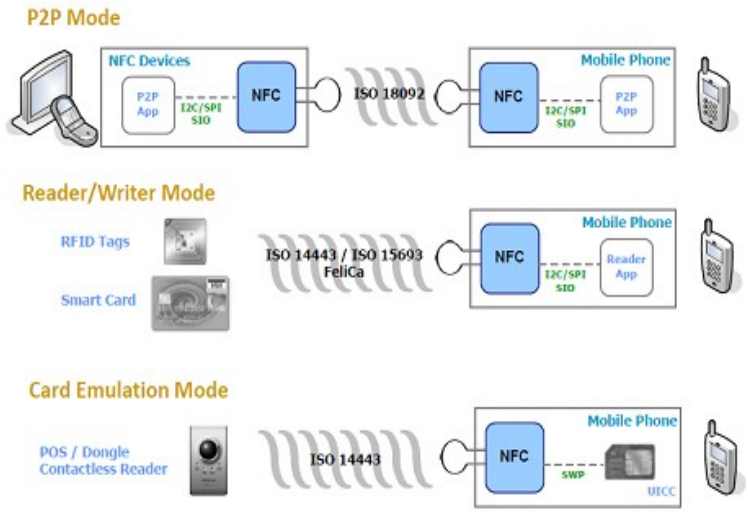
5 *http://www.sonymobile.com/es/products/accessories/smarttags-nt3/*

Cuando el lector se aproxima a una etiqueta o a otro lector, emite una señal de radio de corto alcance que excita al microchip de la etiqueta, con lo que podremos acceder a leer la pequeña cantidad de datos que se encuentran almacenados en ésta. En el caso de la comunicación con etiquetas o *tags,* es el lector el encargado de establecer la comunicación.

En el protocolo NFC siempre hay un dispositivo que inicia la comunicación, siendo este dispositivo el encargado de monitorizarla. NFC permite tres modos de comunicación, que son:

- Punto a punto. Utilizado para el intercambio de datos o establecimiento de las comunicaciones entre dispositivos NFC (utilizando el propio protocolo para unos pocos Kb).
- Lectura-escritura. Tiene la capacidad de leer o escribir etiquetas. Suele utilizarse para los denominados pósters inteligentes (*smart poster*), ya que al leer la etiqueta incluida en el póster, ésta trasmite al teléfono la dirección de una página web, abriendo automáticamente el navegador.
- Emulación de tarjeta. En este modo, el dispositivo NFC se comporta como una tarjeta inteligente, apareciendo ante el lector como una tarjeta sin contacto. Con esta configuración se pueden utilizar las características del elemento de seguridad incorporado como medio de pago, así como para el almacenamiento y gestión de todo tipo de entradas y recibos.

En la siguiente figura[6], puede verse un resumen del funcionamiento de los tres modos de comunicación en NFC.



6 *http://www.myti.it/blog/2013/11/5/nfc-operating-modes*

Toda comunicación NFC consta de 5 partes:

- Descubrimiento: Se efectúa el descubrimiento entre los dos dispositivos.
- Autenticación: Ambos se verifican y se determina si se establece algún tipo de cifrado.
- Negociación: Se determina la velocidad de comunicación y resto de parámetros.
- Transferencia: Se realiza el intercambio de datos.
- Reconocimiento: Se confirma, por parte del receptor la trasferencia de datos.

NFC incluye además un procedimiento de autenticación seguro y un mecanismo anticolisiones[7].

Otro de los aspectos importantes de NFC es su capacidad de interactuar con el dispositivo.

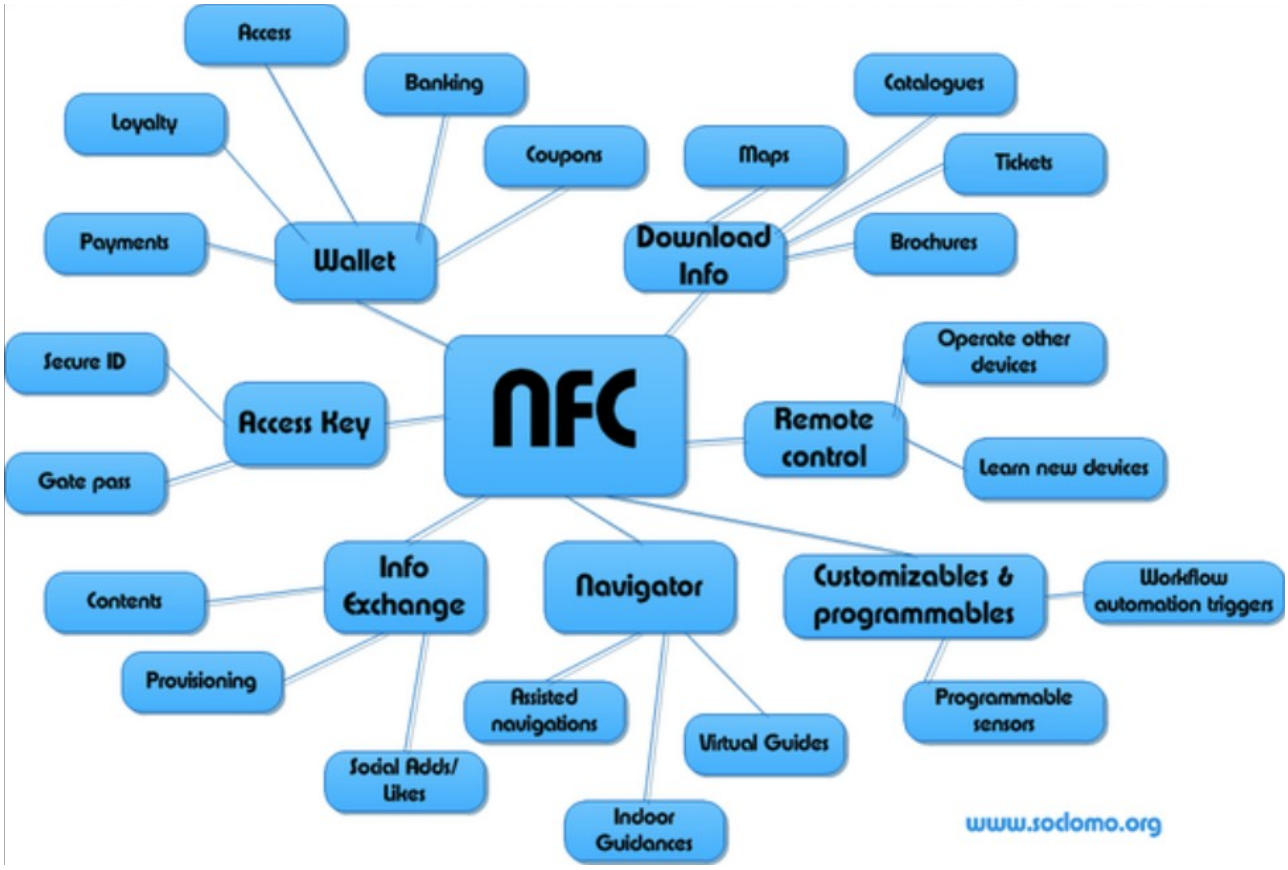
Por ejemplo, una botella de vino con una etiqueta NFC *Smart Poster* podrá proporcionar información detallada del producto a la que está asociada.

Otro aspecto a tener en cuenta en el protocolo NFC es que siempre hay un dispositivo que inicia la comunicación. Este dispositivo se encarga de monitorizar dicha comunicación.

Así pues, con NFC se pueden realizar pagos, proporcionar publicidad mediante *smart posters* y hasta llevar un control de acceso. Gran parte de esta operatividad y versatilidad viene determinada por su corto alcance, lo que la hace inherentemente segura, así como por la rapidez para gestionar una conexión (0'1 seg. frente a los 6 seg. del *Bluetooth*), lo que hace de NFC el medio ideal para la transmisión de pequeñas cantidades de datos.

El siguiente gráfico resume el amplio abanico de funcionalidades que pueden ser utilizadas con esta tecnología.

7 *http://www.gorferay.com/initialization-and-anticollision-iso-iec-14433-3/*


Access
Banking
Loyalty
Coupons
Maps
Catalogues
Tickets
Payments
Wallet
Download Info
Brochures
Secure ID
Access Key
NFC
Operate other devices
Remote control
Gate pass
Learn new devices
Info Exchange
Navigator
Customizables & programmables
Workflow automation triggers
Contents
Provisioning
Assisted navigations
Virtual Guides
Programmable sensors
Social Adds/ Likes
Indoor Guidances
www.sodomo.org

Especificaciones

En el año 2006, "NFC Forum" anunció las primeras especificaciones para NFC:

– NFC *Data Exchange Format* (NDEF).

– NFC *Record Type Definition.*

– NFC *Uniform Resource Identifier Service Record Type Definition.*

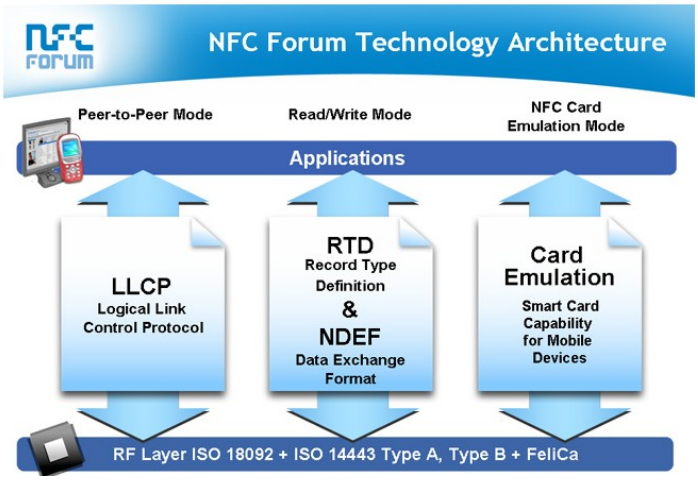
– NFC *Text Record Type Definition.*

– NFC *Smart Poster Record Type Specification*.

La norma base NFC para la capa física es NFCIP-1 (ISO-18092 o ECMA 340), que estandariza la comunicación entre dos dispositivos NFC. El uso de la capa RF (*Radio Frecuency*) en el NFCIP-1 se hereda directamente de las normas más antiguas ISO. En concreto, de la norma ISO-14443 (que define las tarjetas de proximidad sin contacto) y de la norma JIS 6319-4 (norma japonesa en la que se basa el Sony Felica[8]) se hereda el tipo de protocolo que es utilizado por el "NFC Forum" en la etiqueta de tipo 3.

La consecuencia de ello es que los dispositivos NFC (modo de lectura/escritura) son compatibles con la norma ISO-14443 de tarjetas inteligentes.

La arquitectura sobre la que está basada NFC posee tres diferentes modelos de operación: punto a punto, lectura-escritura y emulación de tarjeta, tal y como se muestra en la figura siguiente[9].



8 *http://www.sony.net/Products/felica/*
9 *http://members.nfc-forum.org/aboutnfc/interop/*

LLCP (Logical Link Control Protocol)

LLCP define el protocolo a nivel 2 de la capa OSI, el cual permite soportar comunicaciones punto a punto entre dos dispositivos NFC. Esto es imprescindible para cualquier aplicación NFC bidireccional. La especificación define dos tipos de servicios, sin conexión y orientado a conexión, organizados cada uno de ellos en tres clases de servicio de enlace: sin conexión, orientado a conexión y los dos anteriores a la vez, los cuales se describen brevemente a continuación:

- Servicio sin conexión: Ofrece una conexión mínima con poca fiabilidad y poca o ninguna garantía del control de flujo (difiere en ese punto de las garantías que ofrecen las normas ISO-18092 e ISO-14443).
- Servicio orientado a conexión: Garantizan la entrega de datos, el control de flujo y la multiplexación de la capa de servicios basada en sesiones.

LLCP es un protocolo compacto basado en la norma IEEE 802.2, diseñado para permitir el trasporte de pocos datos en pequeñas aplicaciones o en los protocolos de red como OBEX y TCP/IP. LLCP proporciona también un entorno de servicio más robusto para aplicaciones. Facilita por lo tanto una sólida base para las aplicaciones punto a punto, mejorando la funcionalidad básica ofrecida por la norma ISO/IEC 18092, pero sin impactar en la interoperabilidad de las aplicaciones o integrados NFC.

NDEF (NFC Data Exchange Format)

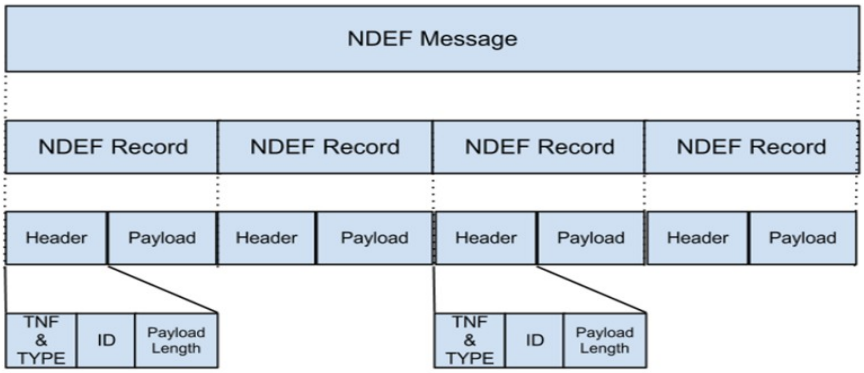
NDEF define el formato del mensaje NFC. Este formato es ligero y se compone de un mensaje que puede contener uno o más registros NDEF. Puede trasportar además una carga útil de hasta 2^{31} octetos de datos. Si fuera necesario enviar más datos, estos mensajes se encadenarían.

El primer registro del mensaje NDEF tiene la marca MB (*Message Begin*), mientras que el último registro tiene la marca ME (*Message End*)

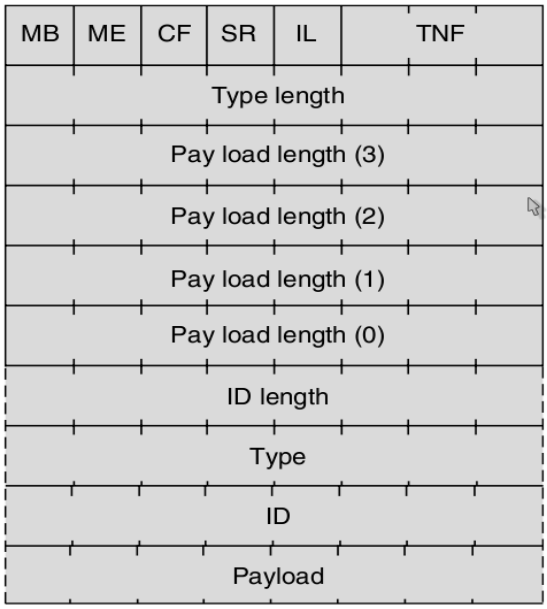
	NDEF message				
R_1 MB = 1	…	…	…	…	R_n ME = 1

Los datos que se encapsulan en el registro NDEF pueden ser de diferentes

tamaños. De esta manera, cada dato enviado se define por un conjunto de tres atributos: la longitud, el tipo y un identificador, tal y como muestra la siguiente imagen[10]:



En la siguiente figura[11] se muestran los ocho primeros bytes que ocupan la cabecera del registro NFC. Contienen las etiquetas que definirán cómo se interpretará el resto del registro.



CF (Chunk Field) Datos fragmentados
SR (Short Record) Si está activado indica que el tamaño está dentro de los limites de los campos de datos (entre 1 y 250 octetos)
IL (ID Length) Activado indica que la cabecera del registro es 1 octeto.
TNF (Type Name Format) Tipo reconocido por NFC

Type_Length Entero sin signo que define la longitud del tipo de campo en octetos.
PayLoad Length Longitud del campo de datos en octetos. Si está activado el bit SR este campo será 1 octeto y si no lo está será 4 octetos.

Type Identifica el tipo de dato. Debe seguir el formato definido en TNF.
ID Contiene una identificación en forma de URI que puede ser absoluta o relativa.
Payload Datos a trasmitir.

Definición del registro NFC (RTD)

La especificación NDEF define el formato común de la información, pero no especifica ninguno en detalle. Estas especificaciones están por separado.

10 *http://sls.weco.net/node/26010*
11 *http://www.cnblogs.com/six-moon/p/5205634.html*

Cada NDEF contiene una cadena con el nombre del tipo de registro, que se denomina RTN (*Records Type Name*). El RTN puede especificar diferentes formatos, como puede ser MIME Types, URIs o tipos NFC conocidos (*The "NFC Forum" Well-known Type*).

Existen dos tipos de espacios de nombres: globales/locales y externos.

Los tipos globales son definidos y gestionados únicamente por "NFC Forum" y se indican con un identificador (NID) con el valor "0x01" en el registro TNF definido por el "RFC 2141". Nadie puede utilizar este identificador para especificar otro espacio de este tipo. Estos nombres deben empezar por mayúscula como "C", "Yfs" o "Viaje-a-Paris". El tipo local puede empezar por número o minúsculas como "1" o "comida".

Los tipos externos son utilizados por empresas que definen un espacio de nombres para sus propias necesidades.

De esta manera, los valores del campo TNF pueden ser:

- 0x01: Tipos NFC Conocidos, especificados en RFC 2234.
- 0x02: Para tipos MIME, especificados en RFC 2046.
- 0x03: URIs absolutas, especificadas en RFC 3986.
- 0x04: Tipos externos, basados en la especificación RTD.
- 0x05: Desconocido, su *type-lengh* (longitud del tipo de datos) debería de ser cero.
- 0x06: Sin cambios, cuando los datos que se mandan están divididos, y por lo tanto en varios mensajes.
- 0x07: Reservado para futuros usos.

Un tipo de nombre conocido tendrá como prefijo de su URN[12] "urn:nfc:wkt:" y con "0x01" en el registro TNF.

Un tipo externo tendrá como prefijo de su URN "urn:nfc:ext:" y TNF="0x04".

De esta manera un "NFC Forum" *Well-Known Type Names* debe construirse cumpliendo los siguientes requisitos (utilizando el formato de lenguaje definido en la RFC 2234):

12 *Uniform Resource Name. Una particular URI definida en el [RFC 2141].*

RTD-URI = "urn:nfc:" nfc-nss
nfc-nss = wkt-nss / external-nss
wkt-nss = wkt-id ":" WKT-type
external-nss = external-id ":" external-type
wkt-id = "wkt" external-id = "ext" WKT-type = local / global
local = (lower / number) *WKT-char
global = upper *WKT-char
external-type = dns-part ":" name-part
dns-part = 1*DNS-char
name-part = 1*WKT-char
WKT-char = upper / lower / number / other
DNS-char = upper / lower / number / "." / "-"
upper = "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
lower = "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x" "y" "z"
number = "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
other = "(" ")" "+" "," "-" "." "=" "@" ";" "$" "_" "!" "*" "'" "."
reserved = "%" "/" "?" "#"

Definición del Registro tipo URI de "NFC Forum"

NFC URI es el tipo de registro que es usado por NDEF para recibir la URI almacenada en una etiqueta NFC de un elemento como pudiera ser un póster inteligente. RTD puede ser considerado como una extensión de un tipo reconocido por NFC. Por ejemplo, si el código es "0x03" y el campo URI contiene "www.unaweb.com", el dispositivo recibirá "https://www.unaweb.com", mientras que si el código fuera "0x04", el dispositivo recibirá "http://www.unaweb.com"

A continuación, la siguiente figura muestra la estructura de un campo URI y una URL como registro URI.

Name	Offset	Size	Value	Description
ID Code	0	1 Byte	URI ID Code	ID code of URI
URI	1	N	UTF-8 String	URI

Offset	Content	Description
0	0×D1	SR=1, TNF=0×01, MB=1, ME=1
1	0×01	Length of record type
2	0×13	Size of payload
3	0×55	NFC well-known record type URI (U)
4	0×01	URI Identifier (http://www.)
5	0×6D 0×6f 0×62 0×69 0×6C 0×65 0×72 0×61 0×64 0×69 0×63 0×61 0×6C 0×73 0×2E 0×63 0×6f 0×6D	String "mobileradicals.com" in UTF 8

Registro tipo Texto

Este registro contiene texto plano y puede ser utilizado en combinación con otros campos para proporcionar una información extra de contenido de la etiqueta.

Como el texto de esta etiqueta puede ser sobrescrito, la especificación recomienda usarlo solo para propósitos informativos, es decir, no vincularlos a ninguna otra acción.

Etiqueta *Smart Poster*

El tipo *Smart Poster* define la capacidad de añadir metadatos[13] a la URI (cosa que RTD no podía hacer). En una etiqueta de *Smart Poster,* al acercar el teléfono (o PDA/Tablet) sobre ella la etiqueta puede proporcionar acciones a realizar embebidas como parte del registro, que desencadenen acciones a realizar por parte del dispositivo.

Los datos enviados a través de NDEF del *Smart Poster* están compuestos por un conjunto de múltiples mensajes de registros NDEF (flag SR=0).

Una etiqueta de *Smart Poster* puede llevar uno o más de los siguientes campos:

- *Title*: Opcional. Puede ser usado más de una vez. Este registro es una instancia de Text_RTD.
- URI: Es una extensión del campo URI añadiendo metadatos a la URI. Forma parte del núcleo del registro de la etiqueta y no puede ser repetido (una sola URI por campo de *Smart Poster*).

13 *Datos ocultos a simple vista en un documento que contienen información adicional del mismo*

– *Action*: Es opcional y se usa para desencadenar acciones como lanzar el navegador, emparejar con un altavoz *bluetooh*, etc.

– *Icon*: Es opcional y se puede usar para incluir una o más imágenes tipo MIME para que el dispositivo las guarde. Por ejemplo para visualizar una imagen en la URI del navegador.

– *Size*: Opcional e indica el tamaño del contenido. Útil cuando se usa en combinación con el tipo de acción.

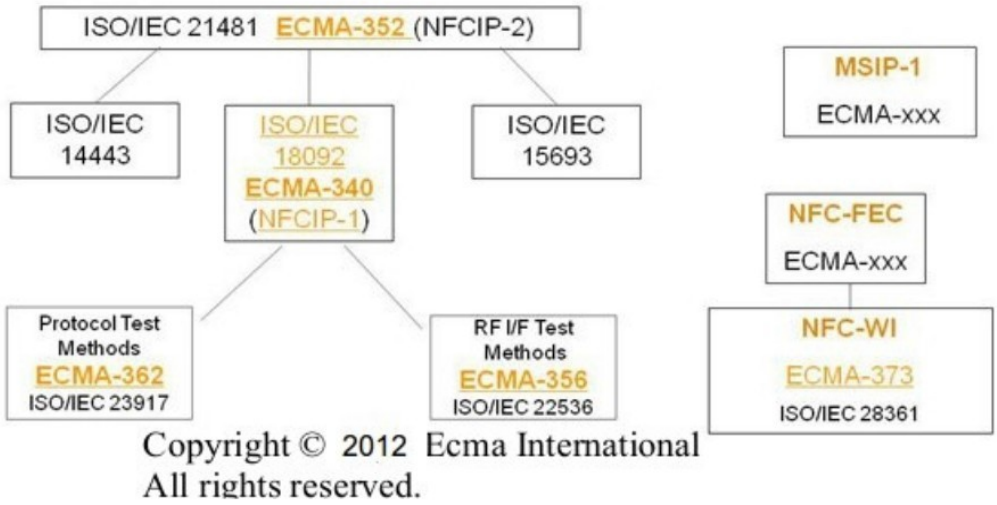
– *Type*: Opcional. Se usa para determinar si el dispositivo está capacitado (en combinación con el campo *size*) para acceder al objeto externo o no.

NFCIP-2

Así como NFCIP-1 describe solamente NFC *Peer-to-Peer*, NFCIP-2 especifica la manera de detectar y seleccionar los siguientes modos de comunicación:

- PCD, *Proximity Coupling Device* (ISO 14443-2, -3 y -4).
- PICC, *Proximity Integrated Circuit Card* (ISO 14443).
- VCD, *Vicinity Coupling Device* (ISO 15693-2,-3).
- NFC (ISO 18092).

De esta manera el estándar NFCIP-2 incluye:

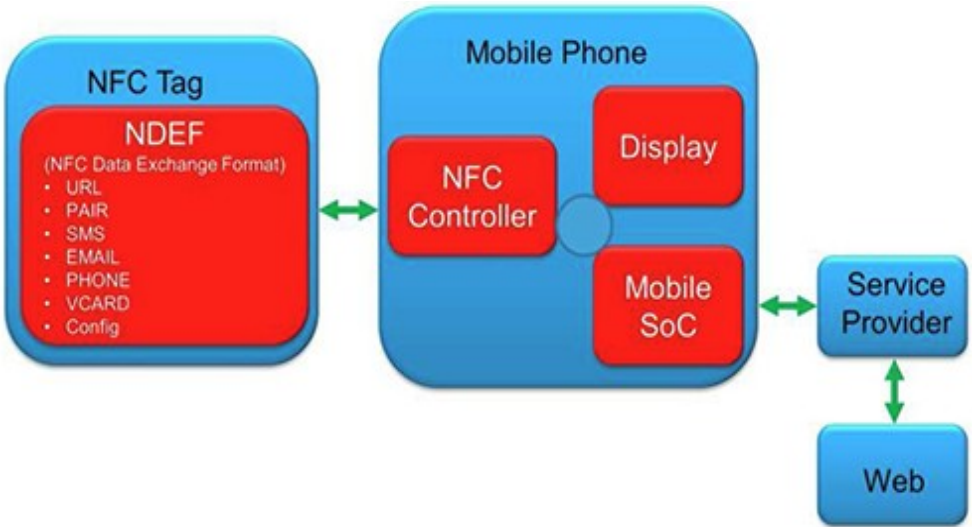


Infraestructura básica:

Como se ha comentado anteriormente, NFC se basa en RFID, por lo que la infraestructura básica utilizada es la misma. La principal diferencia es que en NFC, el emisor es normalmente un teléfono móvil o un *tablet*. Es en este caso cuando el dispositivo utilizado es el que actúa como controlador y provee de energía a las etiquetas para poder leerlas.

El dispositivo que actúa como controlador NFC es el encargado de proporcionar energía mediante un campo electromagnético y así, poder leer las etiquetas.

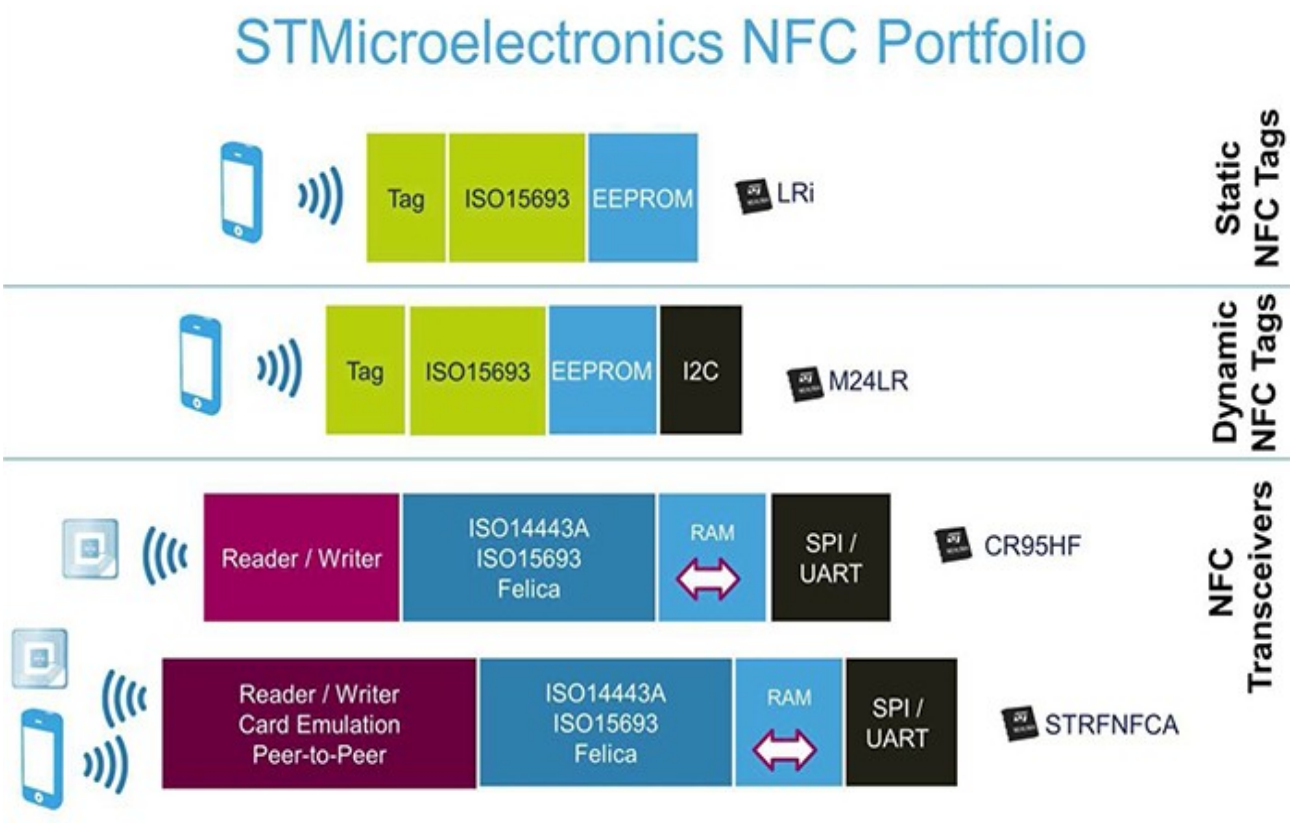
Después de determinarse si la etiqueta puede ser leída, el dispositivo lee el mensaje NDEF (*NFC Data Exchange Format*). Este mensaje está guardado en la EEPROM de las etiquetas y su estructura, que se ha descrito anteriormente, está definida por el "NFC Forum" para proporcionar interoperatividad.



Controlador NFC

Los controladores NFC soportan todos los modos de comunicación NFC y pueden, además, realizar una conexión con un elemento de seguridad para los pagos locales. Esta utilidad está muy extendida en los móviles y tabletas.

La siguiente figura muestra un resumen de las normas y elementos que intervienen en los diversos tipos de interacciones realizadas por dispositivos y elementos NFC.



Especificación técnica del protocolo digital NFC[14]

Esta especificación es la encargada de proporcionar una implementación de las normas ISO/IEC 18092 e ISO/IEC 14443. Armoniza las tecnologías integradas, especifica las opciones de implementación y limita la interpretación de las normas.

En esencia, muestra a los desarrolladores cómo usar la norma ISO/IEC 14443 junto a las normas JIS X6319-4 para garantizar la interoperatibilidad entre diferentes dispositivos NFC, y entre dispositivos NFC y la infraestructura existente.

La especificación define el conjunto común de características que se pueden usar consistentemente y sin modificaciones para aplicaciones NFC, como pueden ser servicios financieros y transportes públicos. La especificación cubre el interfaz digital, y el protocolo de transmisión half-duplex de un dispositivo NFC en sus cuatro roles: Iniciador, Objetivo, Lector/Escritor y Emulador de Tarjeta. La especificación incluye la codificación a nivel de bit, las tasas de bits, formatos, protocolos y conjuntos de comandos utilizados por dispositivos NFC para intercambiar datos y enlazar con el

14 *http://members.nfc-forum.org/specs/spec_list/#tagtypes*

protocolo LLCP.

NFC Especificación técnica de actividad

Explica cómo la especificación técnica del protocolo digital NFC se puede utilizar para configurar el protocolo de comunicación con otro dispositivo o etiqueta NFC. En la especificación técnica de actividad se describen los bloques de construcción, llamados actividades, para configurar el protocolo de comunicación.

Estas actividades se combinan en perfiles y cada perfil tiene los parámetros de configuración específicos, cubriendo un caso de uso particular. En este documento se define cómo se produce el sondeo y el establecimiento de una conexión punto a punto en los dispositivos NFC, la lectura de datos y el formato de intercambio de datos con una etiqueta NFC.

Esta combinación de actividades y perfiles define un comportamiento predecible para un dispositivo NFC. Esto no limita a los dispositivos NFC de la aplicación para utilizar otros bloques de construcción, la definición de otros perfiles o para otros casos de uso por encima de los ya existentes.

Tipos de Etiquetas

Una etiqueta NFC es similar a una etiqueta RFID, existiendo distintos tipos de etiquetas dependiendo de su capacidad, velocidad, etc.

Se puede producir una confusión al leer sobre etiquetas porque en muchos sitios se habla de Tipo A o Tipo B. En esos casos se está haciendo referencia a los estándares ISO-14443 A/B y no a las etiquetas NFC.

A continuación se describen brevemente los distintos tipos de etiquetas NFC[15].

Tipo 1:

Sirven para la mayoría de las aplicaciones NFC.

- Basada en el estándar ISO-14443A.
- Se pueden leer y re-escribir, pudiéndose configurar como solo lectura.

15 *http://apps4android.org/nfc-specifications*

- De 96 bytes hasta 2KB de memoria.
- Velocidad de comunicación de 106 Kbits/s.
- No tiene protección de colisión.
- Compatible con Innovision Topaz, Broadcom BCM20203.

Tipo 2:

Son similares al tipo 1. Derivan de NXP/Philips MIFARE Ultralight tag.

- Incorpora el soporte anticolisión.

Tipo 3:

Deriva del estándar de la industria Japonesa (JIS)X6319-4.

- Memoria variable hasta 1MB.
- Velocidad de comunicación de 214 o 424Kbits/s.
- Compatible con la Felicia de Sony.
- Son más caras que las de tipo 1 y 2.

Tipo 4

Similares a las de tipo 1 y 2, pero derivan de las etiquetas NXP Desfire.

- Totalmente compatible con ISO-14443A/B.
- Preconfiguradas de fábrica como lectura/re-escribible/lectura-escritura.
- Memoria variable hasta 32KB por servicio.
- Soporta velocidades de 106, 212 o 424KBits/s.
- Posee mecanismo de anticolisión.

Tipo 5 (NFC-V)

Permite dar soporte a las etiquetas ISO-15693, aumentando su funcionalidad permitiendo escribir y leer en ellas mensajes en formato *NFC Data Exchange Format.*

Las etiquetas ISO 15693 se han utilizado típicamente en libros, empaquetado, *ticketing* o en empaquetados de medicinas.

MIFARE

MIFARE se refiere a una etiqueta desarrollada por "NXP *semiconductors*"

(www.nxp.com) que es muy usada como tarjeta de memoria en las aplicaciones para el transporte.

- No es una etiqueta "NFC Forum" pero está soportada por muchos dispositivos NFC[16].
- Compatible con NXP MIFARE Classic 1k, MIFARE Classic 4K, y Classic Mini.
- Existen varios tipos:
    - MIFARE Classic: Cumple con la norma ISO-14443 pero incorpora un protocolo propietario para la seguridad y cifrado propiedad de NXP.
    - MIFARE Ultralight: Lleva circuitos integrados de bajo coste. Utiliza el mismo protocolo que la classic pero sin la parte de seguridad.
    - MIFARE Ultraligtht C: proporciona cifrado "triple DES".
    - MIFARE DESFire: Tiene una ROM con un sistema operativo NXP.
    - MIFARE DESFire EV1: incluye cifrado "AES".
    - MIFARE SAM AV2: incluye almacenamiento seguro cifrado.

Felicia

Es una tecnología de etiquetas NFC desarrollada por Sony[17] y ampliamente utilizada para el pago en transportes asiáticos. Tal y como se ha mencionado, son un estándar de la industria japonesa.

Gestión de permisos en una etiqueta MIFARE Classic

A continuación se detallan cómo guarda los permisos una etiqueta MIFARE Classic, que es una de las más utilizadas.

La información se divide en sectores. Cada sector contiene 4 bloques de datos, y a su vez un bloque tiene 16 bytes.

En la imagen siguiente vemos una posible lectura del sector 4 que se correspondería con los bloques 16, 17, 18 y 19. Normalmente éste será su contenido por defecto.

16 *NFC semiconductor es la empresa que fabrica y desarrollo el integrado que incorporan muchos dispositivos NFC*
17 *https://www.sony.net/Products/felica/NFC/relation.html*

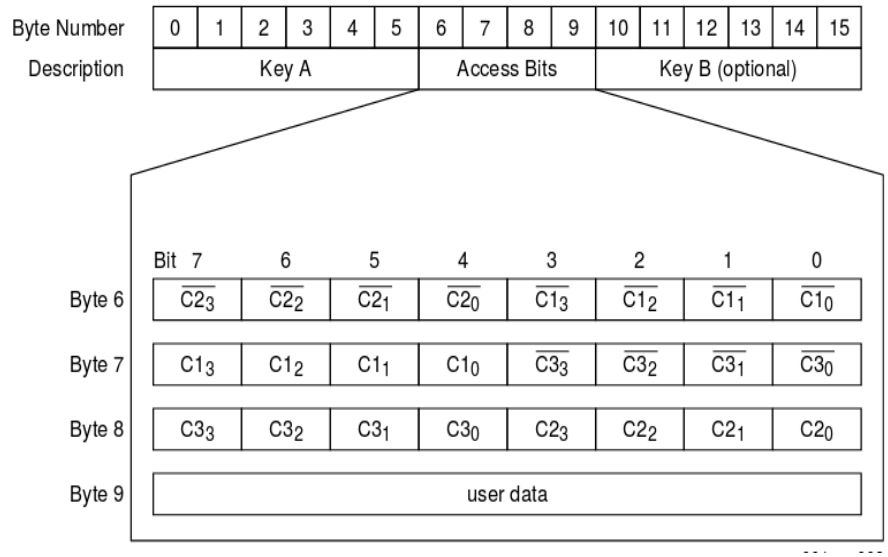
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Bloque 16	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Bloque 17	(datos)
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Bloque 18	
0	0	0	0	0	0	FF	7	80	69	FF	FF	FF	FF	FF	Bloque 19	
Key A						**Access Bits**			*(GP)*	**Key B**					(*Sector Trailer*)	

Observamos que disponemos de los bloques 16, 17 y 18 para guardar información. En caso de que queramos acceder a estos 3 bloques, tenemos que pasar la autenticación del bloque 19, denominado *sector trailer,* y que determinará los permisos guardados. Podemos ver como la información del *sector trailer* está dividida en 3 grupos. Los 6 primeros bytes (0 al 5) contienen la clave A (KEY_A), los 4 siguientes (6 al 9) contienen la gestión del acceso al sector, y los últimos 6 bytes (10 al 15) contienen la clave B (KEY_B).

Por tanto, cada una de las claves tiene 6 valores de 1 byte. Ambas claves suelen tener el valor (FF, FF, FF, FF, FF, FF) por defecto.

> Nota: En el caso de la clave A vemos (0, 0, 0, 0, 0, 0) al leer. Esto se debe a que no tenemos acceso a su lectura y nunca lo tendremos. Es decir, el valor de la clave A, lo tenemos que obtener de otra forma.
>
> Si modificamos los bits de acceso de forma incorrecta, podemos dejar inservible la etiqueta.

El valor de los bytes de la gestión del acceso al sector es, por defecto, (FF, 07, 80, 69). La estructura de estos bytes es a nivel de bits, y conforman una matriz de datos de la siguiente forma.



Se puede ver que a nivel de bits tenemos ternas de celdas:

- Las celdas $C1_0$ $C2_0$ $C3_0$ definen los permisos del bloque 16.
- Las celdas $C1_1$ $C2_1$ $C3_1$ definen los permisos del bloque 17.
- Las celdas $C1_2$ $C2_2$ $C3_2$ definen los permisos del bloque 18.
- Las celdas $C1_3$ $C2_3$ $C3_3$ definen los permisos del bloque 19 (*sector trailer*).

Además vemos que el byte 9 es un byte de propósito general (GPB) que no se usa en la especificación actual[18] (del año 2014). Sin embargo según la especificación anterior[19] (del año 2012), puede contener el número de versión con dos dígitos. Éstos indicarán el número mayor (MN) y menor (mN) de versión de la etiqueta, determinando si se trata de una etiqueta MIFARE Classic o MIFARE Plus tag (llamada MSVNo), y una etiqueta implementada en el dispositivo lector (llamado NFCDevVNo).

La siguiente tabla determinará si el tipo de etiqueta puede ser leída.

nº	número de versión	Significado
1	$NFCDevVNo_{MN} = MSVNo_{MN}$ y $NFCDevVNo_{Mn} >= MSVNo_{Mn}$	Se trata de una MIFARE Plus tag o MIFARE Classic que usa las características generales de este tipo de etiquetas
2	$NFCDevVNo_{MN} = MSVNo_{MN}$ y $NFCDevVNo_{Mn} < MSVNo_{Mn}$	No todas las funcionalidades de MIFARE Classic o MIFARE Plus tag serán accesibles
3	$NFCDevVNo_{MN} < MSVNo_{MN}$	Formato de datos incompatible. No se trata de una etiqueta MIFARE Classic o MIFARE Plus tag
4	$NFCDevVNo_{MN} > MSVNo_{MN}$	Versión anterior de etiqueta. Se podrá leer esta tarjeta si el dispositivo lector lleva implementada dicha funcionalidad.

18 *http://cache.nxp.com/documents/data_sheet/MF1S50YYX_V1.pdf*
19 *http://www.nxp.com/documents/application_note/AN1304.pdf*

Observamos también que las celdas están duplicadas pero con sus celdas homólogas negadas. Esto quiere decir que si queremos poner $C1_0$ con el valor 1, en $/C1_0$ tenemos que poner el valor 0. Si esto no se respeta, este sector puede quedar inutilizable.

A continuación, vamos a ver cómo se guardan los permisos que queremos poner y qué valores debemos poner en los 3 bytes de acceso (6 al 8) para dichos permisos. Para ello vamos a partir de los datos que hemos leído. En el byte 6 tenemos FF. Lo que hacemos es pasar ese número a binario para ver el valor de los bits. Hacemos lo mismo con los bytes 7 y 8, recordando que el byte 9 es de uso libre.

Para el ejemplo anterior tendríamos los siguientes datos.

		$/C2_3$	$/C2_2$	$/C2_1$	$/C2_0$	$/C1_3$	$/C1_2$	$/C1_1$	$/C1_0$
Byte 6	FF	1	1	1	1	1	1	1	1
		$C1_3$	**$C1_2$**	**$C1_1$**	**$C1_0$**	$/C3_3$	$/C3_2$	$/C3_1$	$/C3_0$
Byte 7	07	**0**	**0**	**0**	**0**	0	1	1	1
		$C3_3$	**$C3_2$**	**$C3_1$**	**$C3_0$**	**$C2_3$**	**$C2_2$**	**$C2_1$**	**$C2_0$**
Byte 8	80	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Byte 9	...	*.... da igual ....*							

Vemos cómo para el bloque 16 tenemos permisos C1=0, C2=0 y C3=0 los mismos que para los bloques 17 y 18. De esta manera, tendremos los mismos permisos para los 3 bloques de datos. En cambio tenemos que para el bloque 19, que es el *sector trailer*, C1=0, C2=0 y C3=1.

Para interpretar estos resultados, tenemos que ir a las siguientes tablas[20] que son las que nos da el estándar de las etiquetas Mifare 1k.

20 *http://cache.nxp.com/documents/data_sheet/MF1S50YYX_V1.pdf*

Tabla de condiciones de acceso para el *sector trailer*

Access bits			**Access condition for**						**Remark**
			KEYA		**Access bits**		**KEYB**		
C1	C2	C3	read	write	read	write	read	write	
0	0	0	never	key A	key A	never	key A	key A	Key B may be read
0	1	0	never	never	key A	never	key A	never	Key B may be read
1	0	0	never	key B	key A|B	never	never	key B	
1	1	0	never	never	key A|B	never	never	never	
0	0	1	never	key A	key A	key A	key A	key A	Key B may be read, transport configuration
0	1	1	never	key B	key A|B	key B	never	key B	
1	0	1	never	never	key A|B	key B	never	never	
1	1	1	never	never	key A|B	never	never	never	

Tabla de condiciones para los bloques de datos

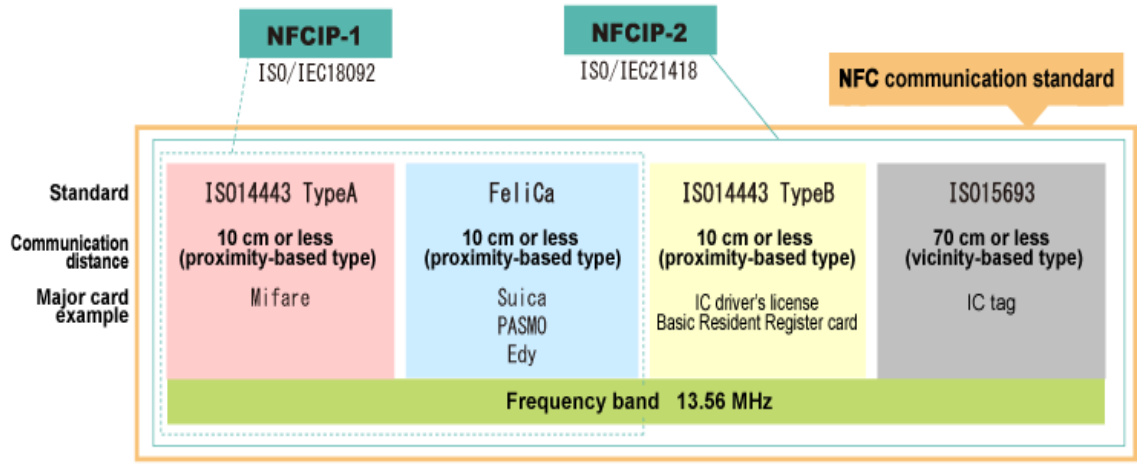
Access bits			**Access condition for**				**Application**
C1	C2	C3	read	write	increment	decrement, transfer, restore	
0	0	0	key A|B	key A|B	key A|B	key A|B	transport configuration
0	1	0	key A|B	never	never	never	read/write block
1	0	0	key A|B	key B	never	never	read/write block
1	1	0	key A|B	key B	key B	key A|B	value block
0	0	1	key A|B	never	never	key A|B	value block
0	1	1	key B	key B	never	never	read/write block
1	0	1	key B	never	never	never	read/write block
1	1	1	never	never	never	never	read/write block

De esta manera tenemos permisos para leer y escribir en los bloques de datos tanto si nos autentificamos con la KEY_A o con la KEY_B, pero sólo tenemos permiso para cambiar la KEY_A, cambiar los permisos, leer la KEY_B o cambiar la KEY_B si nos autentificamos con la KEY_A. Esto es lo que significan los valores [FF , 07, 80].

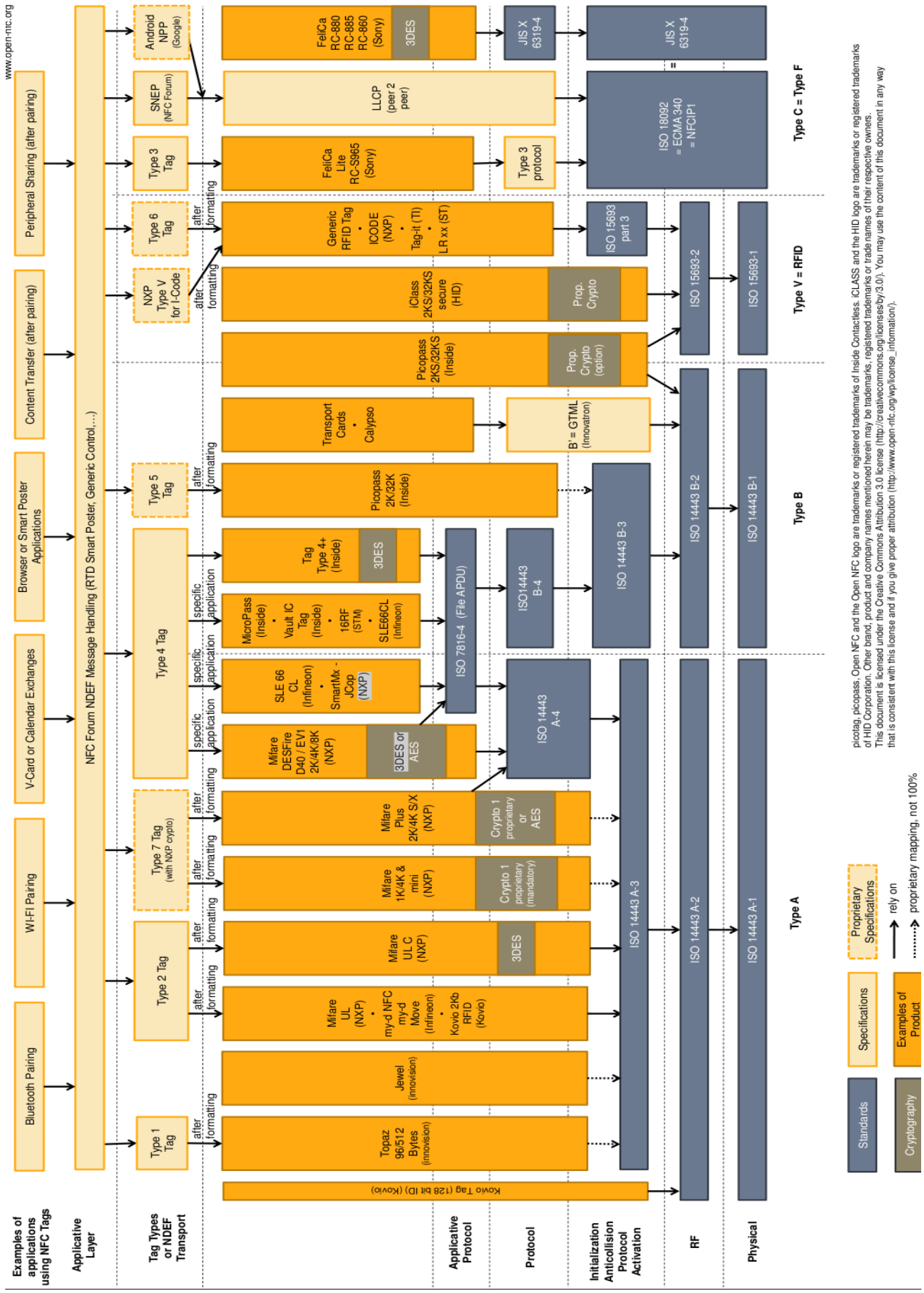
Resumen de las características técnicas de NFC.

	NFC devices		Contactless Smart Card systems	
	Initiator	**Target**	**Reader/writer**	**Contactless smart cards**
Communication principle	**Active communication mode:** Both initiator and target generate RF field **Passive communication mode:** Initiator generates RF field and target answers using load modulation		Reader generates RF field and card answers using load modulation	
Initialization	**Active communication mode:** RF collision avoidance **Passive communication mode:** Initialization and Anticollision		Initialization and anticollision	
Speed at initialization [kbit/s]	106, 212, 424		106 for ISO 14443-A 212, 424 for FeliCa	
Communication protocol	NFC IP-1 data exchange protocol		ISO 14443 Transmission protocol MIFARE®: fixed command set FeliCa™: fixed command set	
Speed at Communication protocol[kbit/s]	106, 212, 424		106, 212, 424, 848 for ISO 14443-A incl. amendments 106 for MIFARE® 212 for FeliCa™	

Los estándares NFC en la actualidad son:



La tabla de la siguiente página expone la relación entre las etiquetas y los protocolos que utilizan.



Seguridad en NFC

NFC se está convirtiendo en una herramienta omnipresente en numerosas acciones que se realizan de forma cotidiana, como el pago de transporte público, el uso en las tarjetas de crédito o acceso a instalaciones. Los estándares de seguridad de NFC proporcionan una base fiable para que dichas acciones se realicen correctamente y evitar que terceras personas obtengan información privada.

Sin embargo, debido a que las comunicaciones a nivel de enlace no están cifradas, la tecnología NFC no está exenta de ataques. Incluso parece abonada a ellos. La mayoría de estos ataques usan su principal característica, la usabilidad, es decir, la transparencia que ofrece al usuario puede ser usada para atacarle. Y éste, junto a las limitaciones del dispositivo NFC (móvil o tarjeta), dificultan una solución sobre todo debido al reducido tamaño del dispositivo en el que se deben alojar.

De entre los diferentes tipos de ataques, destaca el denominado *Relay-Atack*, utilizado para redirigir la información de un dispositivo a otro. Este ataque es fácil de evitar realizando una redundancia en la comprobación (autenticación doble) a la hora de realizar las transacciones, mediante la utilización de certificados para firmar las etiquetas.

Otro ataque posible es la puesta en marcha de servicios sin el consentimiento del usuario. De esta manera podría activarse el *bluetooh* y acceder a ficheros en el dispositivo con permisos totales. El detalle puede verse en el ataque, aunque antiguo, al Galaxy S3 vía NFC[21].

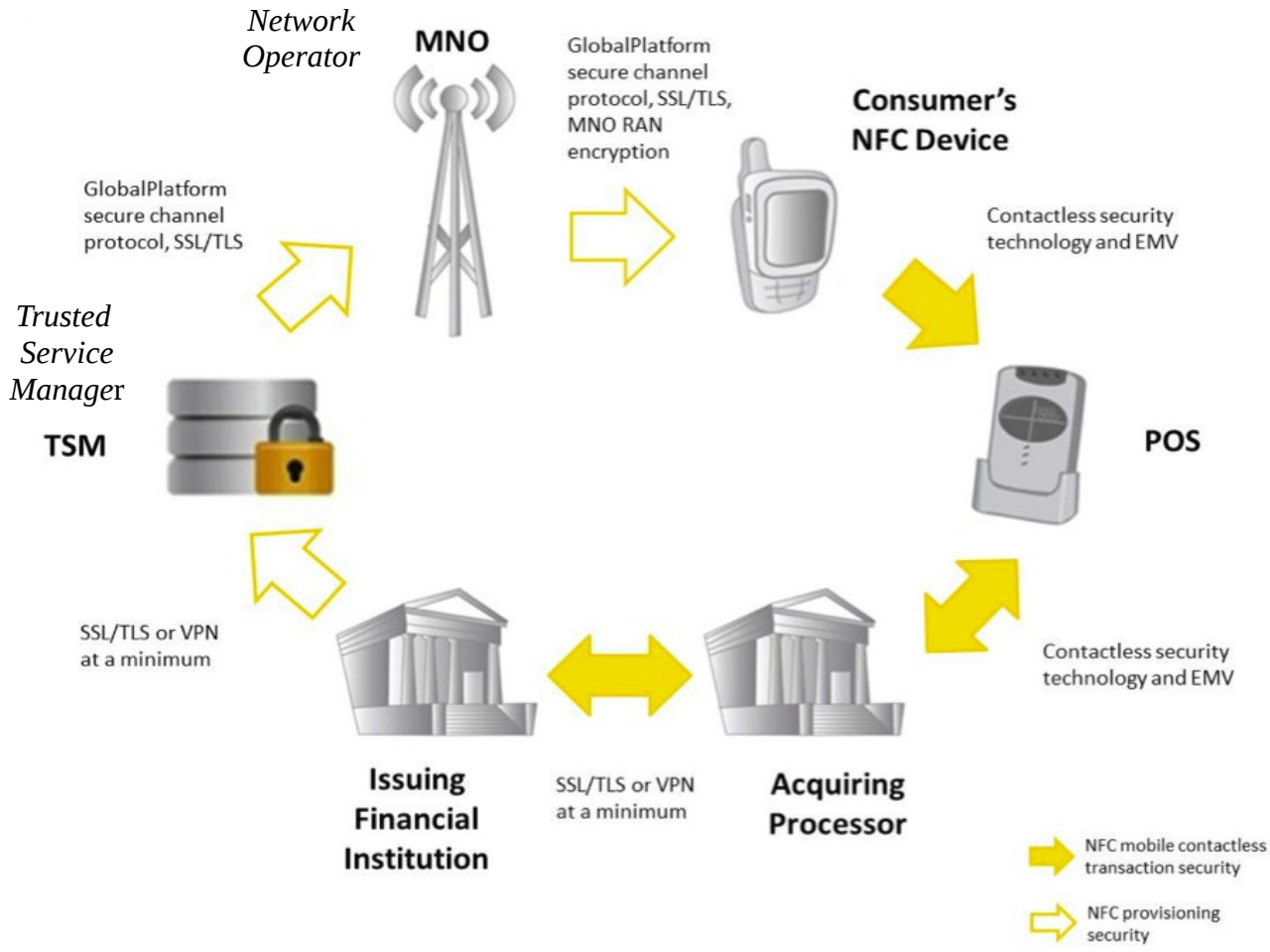
El resto de ataques conocidos tales como *Man in the middle*, Modificación y Corrupción de datos, Denegación de servicio o *Snnifing* son difíciles de realizar debido a la proximidad en la que se realizan las operaciones, ya que recordamos que el radio de acción de esta tecnología es de como máximo 10 cm.

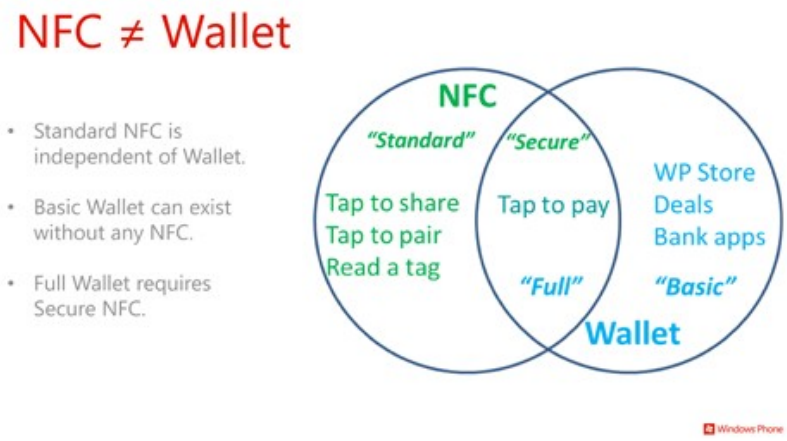
Aparte de éstos, un ataque de *Phising* sí que es posible, ya que una etiqueta puede presentar una URL válida pero enviar una distinta o facilitar una conexión a un punto de acceso malicioso. En la misma línea estaría la "clonación" de etiquetas.

21 *http://www.computerworld.com/article/2492706/cyberwarfare/galaxy-s3-hacked-via-nfc-at-mobile-pwn2own-competition.html*

En cuanto a la seguridad del pago mediante NFC, su seguridad radica en las mismas bases que están siendo usadas en la actualidad para las tarjetas de crédito, pero principalmente sobre una, la confianza. Cuando nosotros (o un empleado del comercio) pasa la tarjeta, ante todo confiamos en que esa transacción es segura y fiable, entonces, ¿por qué hacerlo con el móvil no ha de serlo?.

La siguiente figura representa el mecanismo de seguridad para el pago con dispositivos móviles.





Otro problema existente es que muchas aplicaciones no guardan su información cifrada. Esto hace posible que en el caso de que un atacante se hiciera con el control del teléfono y tuviera por lo tanto acceso con permisos completos para abrir ficheros, podría por leer y modificar el contenido de los mismos.

Para proporcionar una seguridad añadida, NFC cuenta con un chip especializado. En ese camino está Samsung con su nuevo chip "Secu-NFC"[22], que tiene además una pequeña memoria encargada de almacenar la información más sensible sobre datos de pago, contraseñas y cifrado de los datos. NXP[23], una de las empresas más importante de semiconductores, también ha integrado un chip de seguridad NFC en numerosos dispositivos.

Para terminar, empresas como Orange/Vodafone van a introducir o han introducido esta tecnología en sus nuevos abonados, incluyendo el chip NFC en la tarjeta SIM, y haciendo posible que con cualquier teléfono se pueda usar esta tecnología.

Otras empresas, como la Caixa, han realizado experiencias piloto de cajeros con NFC en Mallorca, pensando en la introducción progresiva de esta tecnología en sus cajeros automáticos.

Sin embargo, la comunidad NFC sigue avanzando en aportar soluciones a la seguridad, en Junio del 2015, la *Ecma General Assembly*[24] aprobó los siguientes estándares:

22 *https://news.samsung.com/global/samsung%E2%80%99s-new-secu-nfc-chip-enables-secure-mobile-payment-by-enhancing-nfc-with-security-features*
23 *Htto://www.nxp.com (NXP Semiconductors)*
24 *http://www.ecma-international.org/memento/index.html*

ECMA-385 4ª edición - NFC-SEC: NFCIP-1 Servicios seguros y protocolos.

ECMA-386 3ª edición - NFC-SEC-01: Cifrado con ECDH y AES.

ECMA-409 2ª edición - NFC-SEC-02: Cifrado usando ECDH-256 y AES-GCM.

ECMA-410 2ª edición - NFC-SEC-03: NFC-SEC Autenticación usando Cifrado de clave asimétrica.

ECMA-411 2ª edición - NFC-SEC-04: NFC-SEC Autenticación usando Cifrado de clave simétrica.

Especificaciones técnicas

ECMA-385 especifica un canal NFC-SEC seguro para NFCIP-1 y los dispositivos portátiles.

Los servicios que ofrece para implementar dicho canal seguro son el SSE (*Shared Secret Service*) y SCH (*Secure Channel Service*), de acuerdo con el cifrado NFC-SEC para definir el PID (*Private Indentifier*). Un esquema de la localización de los servicios de seguridad en NFC se puede ver en la figura siguiente.

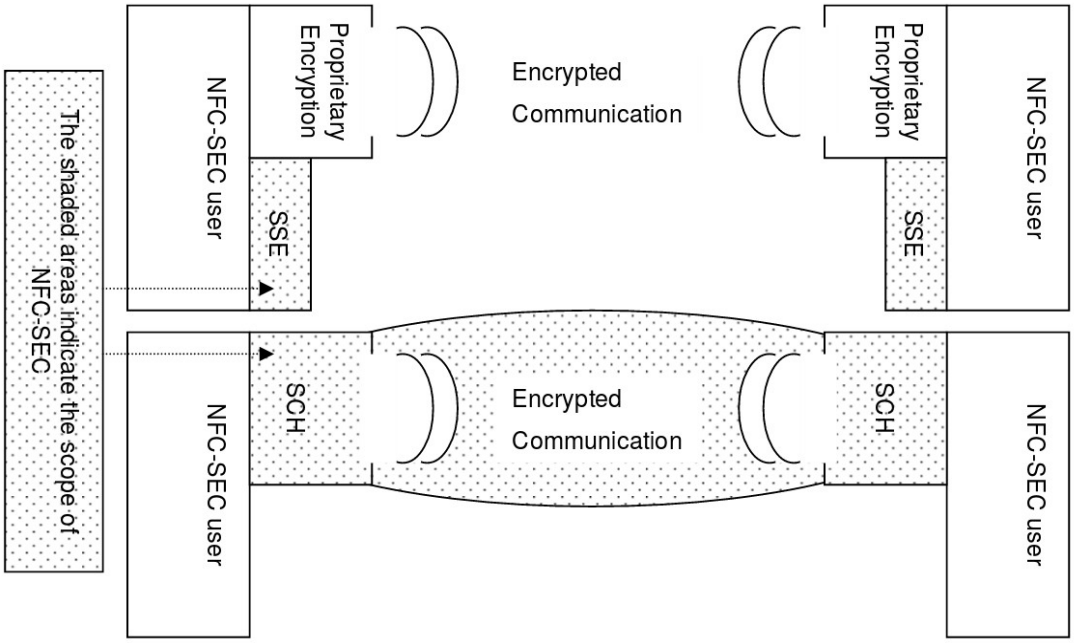
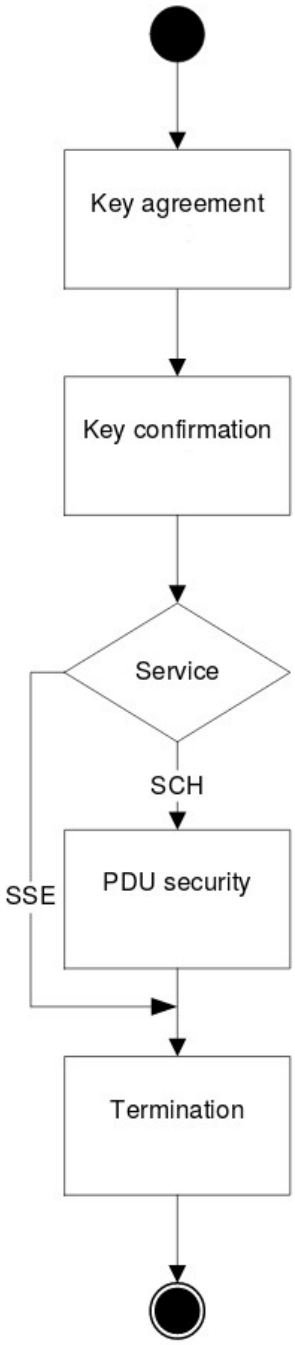


Diagrama de flujo del servicio NFC-SEC

A continuación, en la figura siguiente, se puede ver el diagrama de flujo del servicio NFC-SEC.



Se establece la clave usando ACT_REQ y ACT_RES, conforme a NFC-SEC y se define el PID.

Se verifica la clave usando VFY_REQ y VFY_RES.

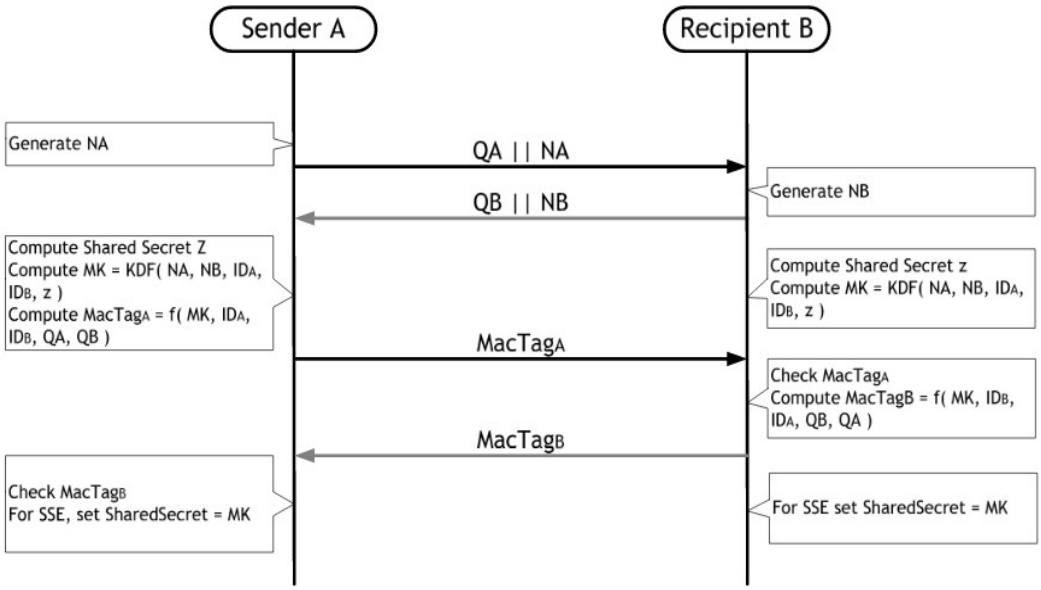
Uno o más de los siguientes pasos

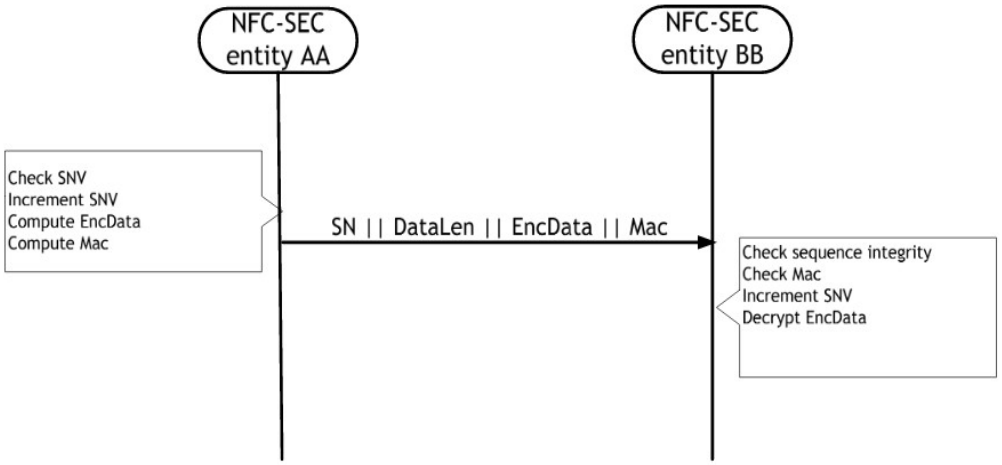
- Secuencia de integridad
- Confidencialidad.
- Integridad de datos.
- Autenticación de origen.

Se termina SSH y SCH, terminando el protocolo de unidad de datos según ISO/IEC 7498-1

ECMA-386 especifica los mecanismos criptográficos que usan un protocolo de curvas elípticas de Diffie-Hellman (ECDH)[25] para las claves, y un algoritmo AES[26] para la integridad y el cifrado de datos. La tercera edición asegura el uso de los últimos estándares criptográficos.

En los siguientes diagramas de secuencia se muestran cómo se establece y comparte la clave secreta (además de la confirmación de la misma) y el protocolo de intercambio de datos de dos dispositivos mediante NFC-SEC según especifica la ECMA-385.



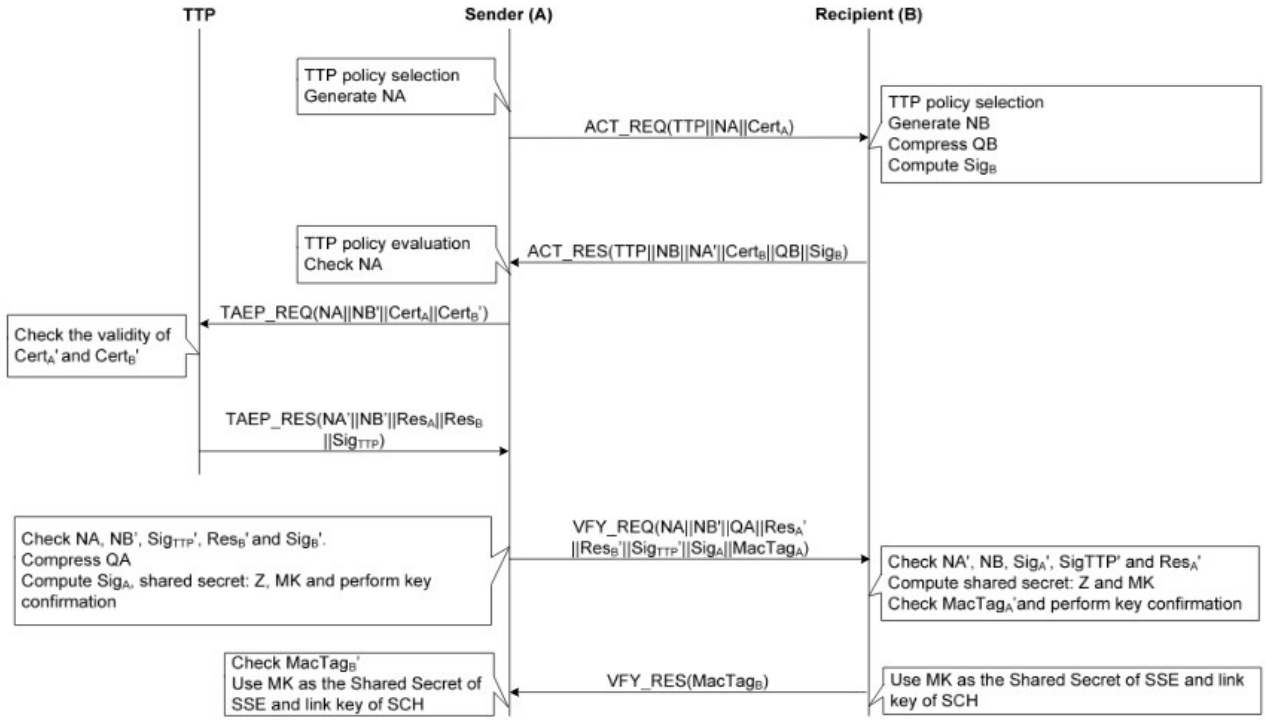


25 *https://msdn.microsoft.com/es-es/library/cc488016(v=vs.90).aspx*
26 *https://es.wikipedia.org/wiki/Advanced_Encryption_Standard*

Por otra parte, ECMA-409 especifica los mecanismos criptográficos que usan las curvas elípticas de Diffie-Hellman con una clave de longitud de 256 bits para la clave y el algoritmo AES en modo Galois/Counter (GCM)[27] para la autenticación. En esta segunda edición se introduce la referencia al último estándar ISO/IEC JTC1/SC27 para generar diferencias entre cada mensaje de acuerdo a la norma ISO/IEC 19772:2009/ Cor.1:2014 que también cumple con NIST SP 800-38B[28].

ECMA-410 especifica cómo se negocian la clave y los mecanismos para la mutua autenticación, el uso de cifrado asimétrico y los requisitos del protocolo para el intercambio entre el emisor y una tercera parte, involucrada en la autenticación. En su segunda edición introduce los estándares JTC 1/SC 27[29], incluyendo ISO/IEC 9798-3, que especifica los mecanismos de confianza on-line cuando está involucrada una tercera entidad (TTP).

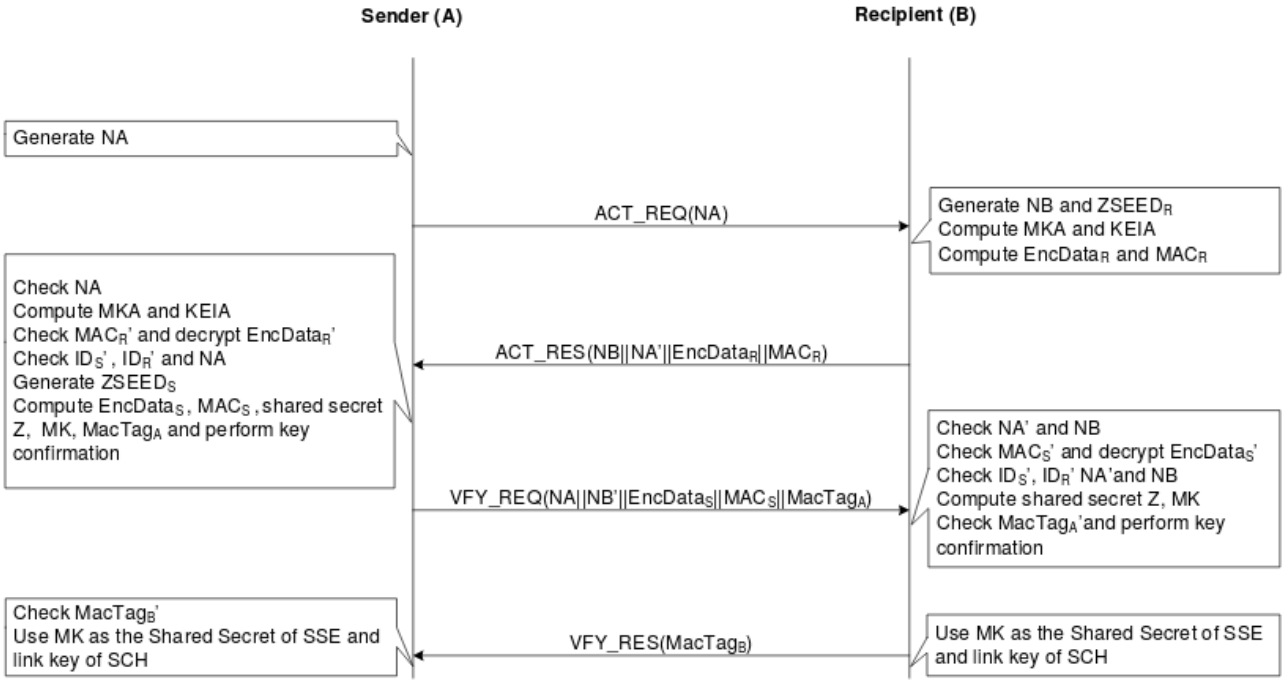
El siguiente diagrama de secuencia ilustra cómo se realiza esta acción.



27 *https://en.wikipedia.org/wiki/Galois/Counter_Mode*
28 *Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication*
29 *http://www.iso.org/iso/jtc1_home.html*

ECMA-411 especifica los mecanismos para obtener la clave y la autenticación mutua usando cifrado simétrico. Esta segunda edición también introduce la diferencia entre mensajes en NFC-SEC-02.

El siguiente diagrama de secuencia ilustra cómo se produce la autenticación NEAU-S (*Secure NFC Entity Authentication*).



Estas nuevas normas serán enviadas al comité técnico de la ISO/IEC JTC 1[30] para su aprobación como estándares internacionales en virtud del procedimiento abreviado ISO/IEC. Las versiones anteriores de ECMA-385 y ECMA-386 ya están disponibles como ISO/IEC 13157-1 e ISO/IEC 13157-2, respectivamente.

30 *http://www.iso.org/iso/jtc1_home.html*

Expectativas

Una de las principales ventajas de NFC es su rapidez de respuesta. En la siguiente tabla tenemos una comparativa con *Bluetooth*;

	NFC	Bluetooth
Compatibilidad RFID	ISO 18000-3	no
Estándar	ISO/IEC	Bluetooth SIG
Estándar de red	ISO 13157, etc.	IEEE 802.15.1
Tipo de red	Punto a punto	Punto a multipunto
Rango	~ 10 cm	~10 m (clase 2) ~100m (clase 3)
Frecuencia	13.56 MHz	2.4-2.5 GHz
Tasa de transferencia	424 Kbit/s	2.1 Mbit/s (v2.1) (mayor a 721 Kbit/s)
Tiempo de inicialización	< 0.1 s	~ 6 s
Seguridad	Sí, a nivel hardware y protocolo	Sí, a nivel protocolo
Modos de comunicación	Activo-pasivo Activo-activo	Activo-activo

Básicamente, NFC cuenta con un menor rango de alcance, pero a su vez presenta un mejor tiempo para establecer una conexión de datos. Esto quiere decir que esta tecnología es la ideal para transmitir pequeños paquetes de datos en distancias cortas. Basta con colocar el dispositivo NFC frente a la etiqueta, y la transferencia de datos se efectuará casi al instante.

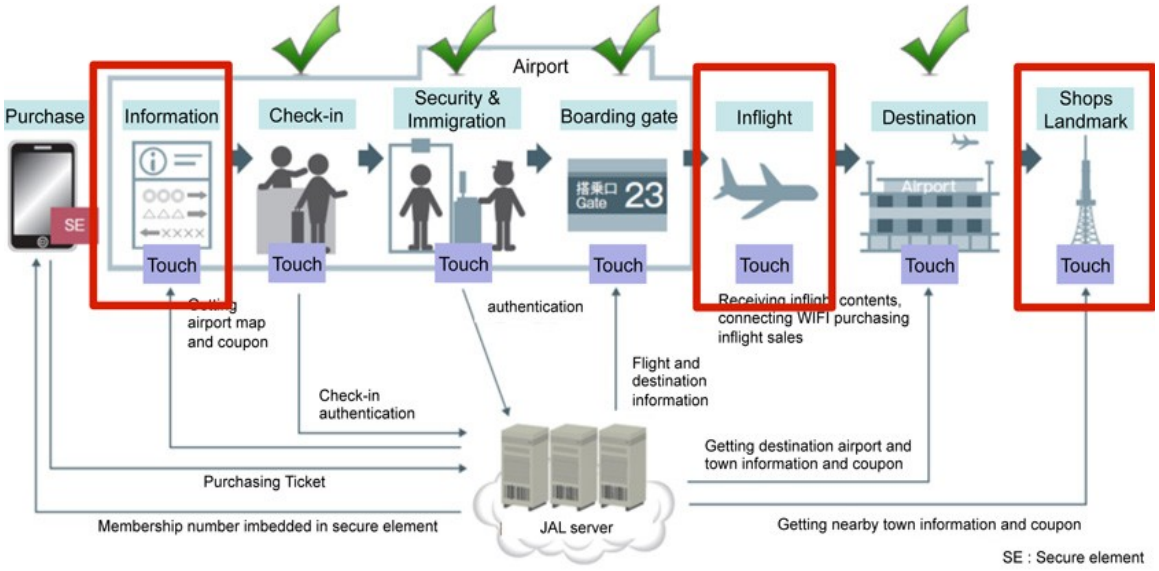
Lo normal es que la comunicación necesite de terceros agentes para poder ofrecer una mayor funcionalidad.

El siguiente ejemplo representa ese caso. En él, un usuario adquiere un billete electrónico de un modo seguro. Una vez en el aeropuerto, ese billete electrónico se va validando en distintos puntos (representados con *touch* en la imagen).

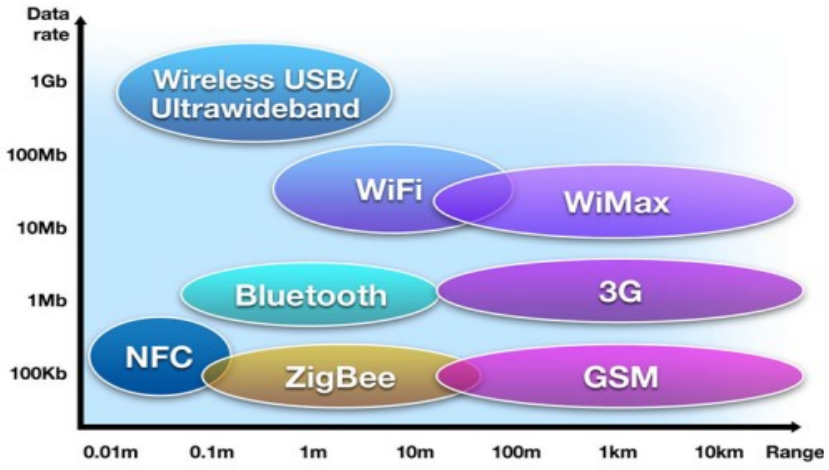
En el punto de control que representa el *check-in,* ese billete electrónico es verificado por un agente externo al aeropuerto que valida su autenticación.

En el resto de los sitios recibirá información relacionada o de interés proporcionada por el servicio (ofertas, planos, detalle del vuelo, etc.).

En cualquier caso, en todos los puntos, solo tendrá que acercar el teléfono al terminal o a una etiqueta NFC.



A continuación, en el siguiente gráfico se comparan las tecnologías inalámbricas actuales para ubicar dónde se encuentra NFC. Como se puede ver, NFC es la tecnología que transmite la menor cantidad de datos en la menor distancia.



Recordamos que existen dos modos de funcionamiento en NFC:

- Pasivo: en las que un solo dispositivo genera el campo electromagnético, mientras que el otro se aprovecha de la modulación para poder transferir los datos. En este caso el dispositivo que inicia la comunicación es el que genera dicho campo.
- Activo: ambos dispositivos generan campos electromagnéticos, puesto que ambos dispositivos necesitan energía.

Si tomamos el ejemplo de usar NFC para efectuar pagos, la comunicación del

teléfono y un cajero será activa. Por otro lado, la de un punto de venta y el teléfono sería pasiva.

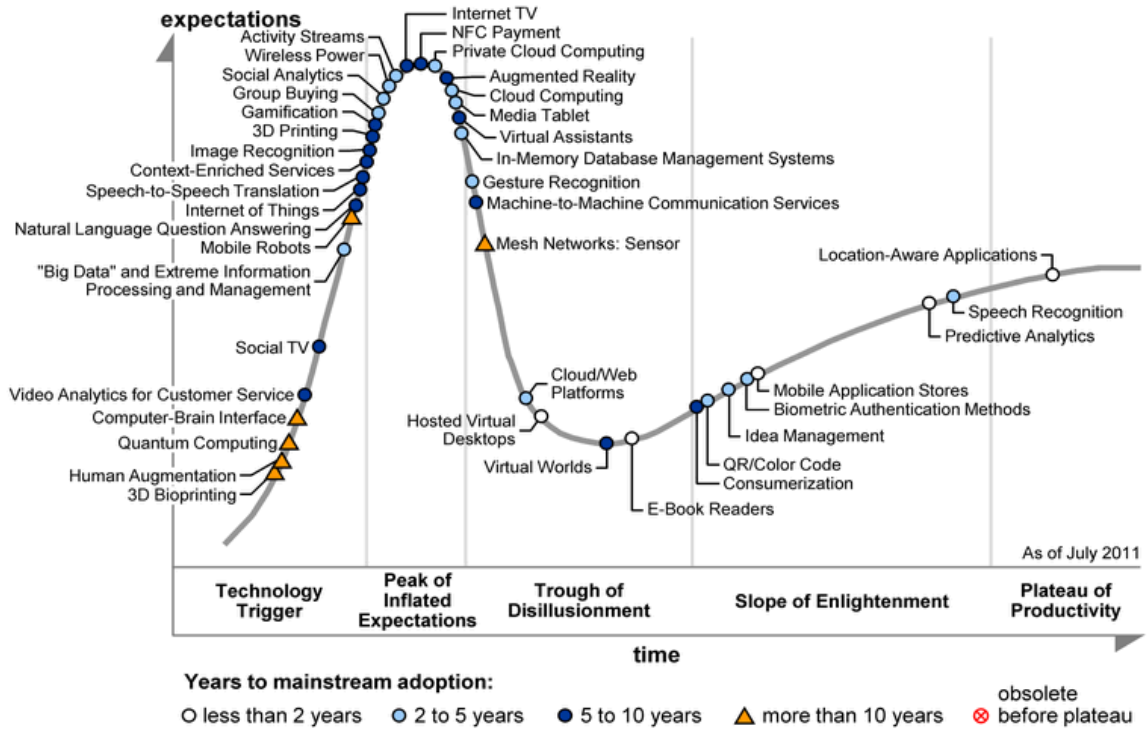
¿Hacia dónde va NFC?

NFC es en la actualidad una tecnología poco conocida para el público general, aunque algunas de sus funcionalidades están siendo implantadas en tecnologías que ya se están utilizando y que han sido mayoritariamente aceptadas como son las tarjetas de crédito, los teléfonos móviles inteligentes (*smartphones*) y las tarjetas de transporte público. De hecho, el primer dispositivo con NFC, el Nokia 6131, salió al mercado en Abril del 2006.

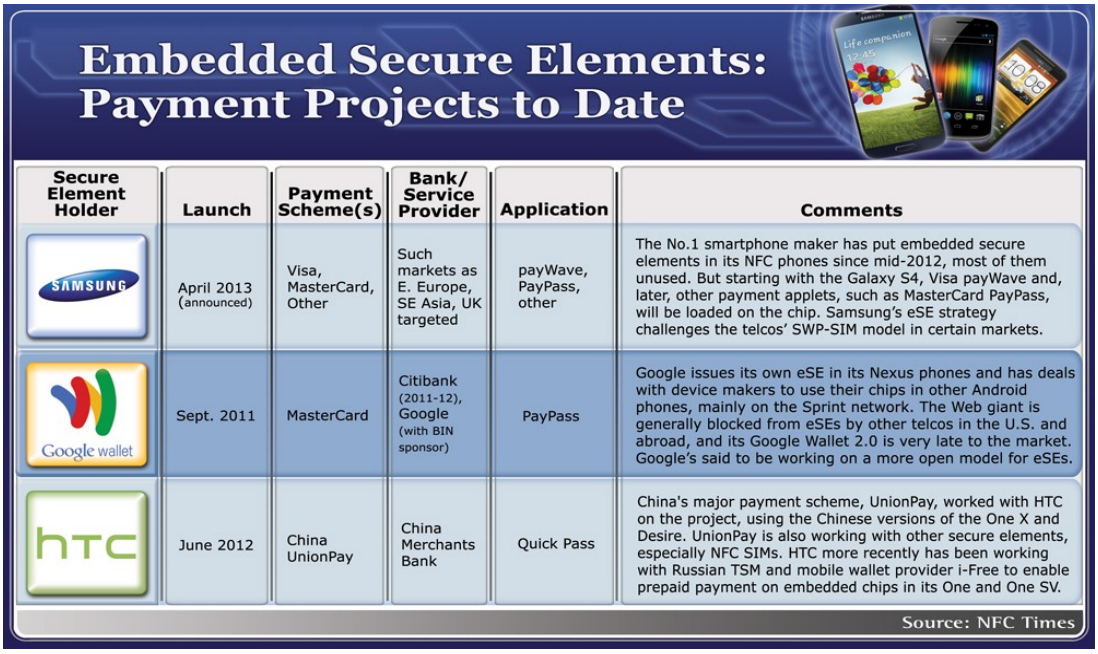
Dado el número y la importancia de las empresas involucradas en NFC, el futuro de esta tecnología parece garantizado. En el gráfico siguiente se pueden ver algunas de estas empresas[31] (alguna sorprende).

31 *http://nfc-forum.org/about-us/our-members/*

Sirva el siguiente gráfico de las diferentes tecnologías que se están desarrollando y ofreciendo en estos momentos para observar cómo las expectativas previstas en 2011 para NFC se han cumplido:



En el siguiente resumen podemos ver cómo ha evolucionado la inclusión de NFC en el mercado de pago.

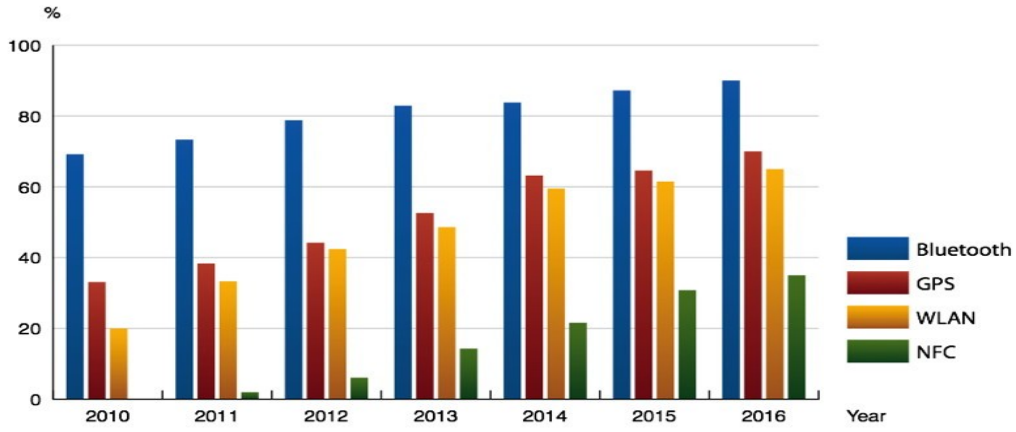


Embedded Secure Elements: Payment Projects to Date

Secure Element Holder	Launch	Payment Scheme(s)	Bank/ Service Provider	Application	Comments
SAMSUNG	April 2013 (announced)	Visa, MasterCard, Other	Such markets as E. Europe, SE Asia, UK targeted	payWave, PayPass, other	The No.1 smartphone maker has put embedded secure elements in its NFC phones since mid-2012, most of them unused. But starting with the Galaxy S4, Visa payWave and, later, other payment applets, such as MasterCard PayPass, will be loaded on the chip. Samsung's eSE strategy challenges the telcos' SWP-SIM model in certain markets.
Google wallet	Sept. 2011	MasterCard	Citibank (2011-12), Google (with BIN sponsor)	PayPass	Google issues its own eSE in its Nexus phones and has deals with device makers to use their chips in other Android phones, mainly on the Sprint network. The Web giant is generally blocked from eSEs by other telcos in the U.S. and abroad, and its Google Wallet 2.0 is very late to the market. Google's said to be working on a more open model for eSEs.
htc	June 2012	China UnionPay	China Merchants Bank	Quick Pass	China's major payment scheme, UnionPay, worked with HTC on the project, using the Chinese versions of the One X and Desire. UnionPay is also working with other secure elements, especially NFC SIMs. HTC more recently has been working with Russian TSM and mobile wallet provider i-Free to enable prepaid payment on embedded chips in its One and One SV.

Source: NFC Times

Por último, en la siguiente gráfica se indica cómo la tecnología NFC se ha ido incorporando a los teléfonos móviles en los últimos seis años.

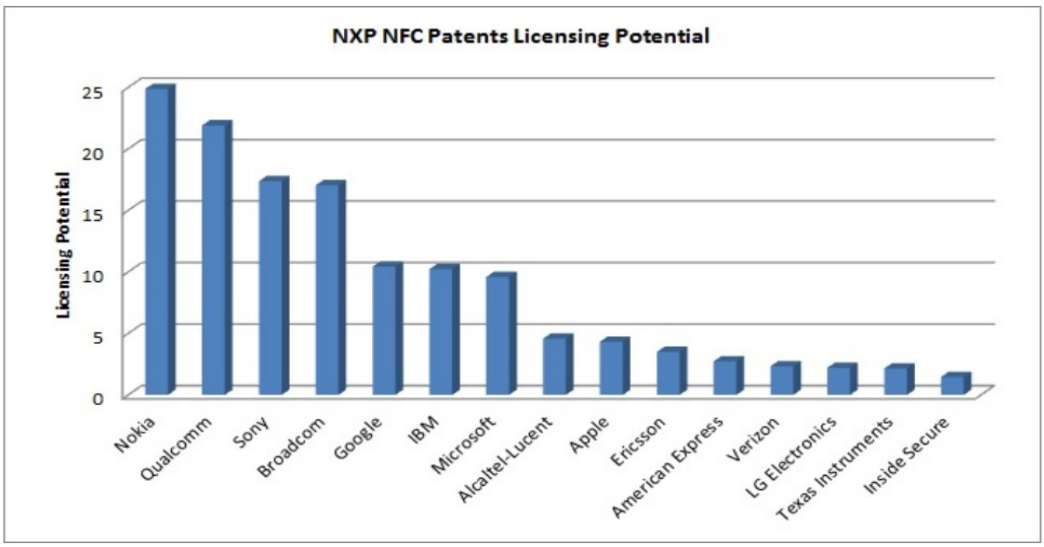


Projected attach rate for handset connectivity technologies (World 2010 - 2016)

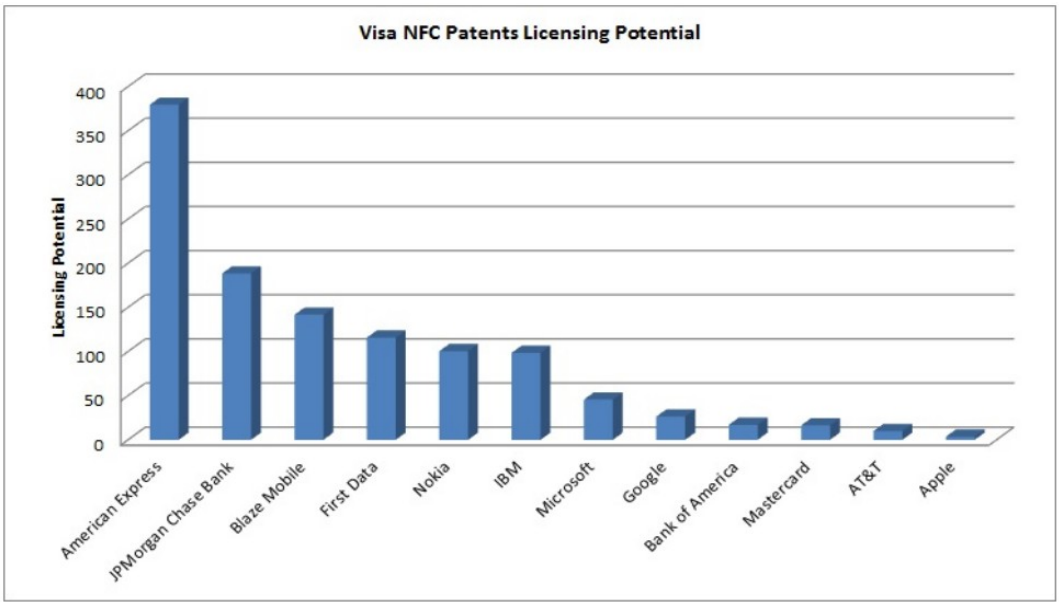
Además cabe destacar que las empresas con patentes en esta tecnología indican que el uso de NFC está en aumento.

Estas patentes de NFC están clasificadas en dos grupos: Tecnología de NFC estándar (ISO y "NFC Forum") y aplicaciones NFC.

El primer grupo incluye el análisis de actividad, protocolos, LLCP (*Logical Link Control Protocol*), NDEF (*Data Exchange Format*), RF/Analog, RTD (*Record Type Definition*) y operaciones con etiquetas. En la siguiente figura se puede ver el número de patentes de las principales empresas involucradas en esta tecnología.



El segundo grupo abarca el análisis del dispositivo NFC y el sistema (como la seguridad), pago con móviles o aplicaciones de negocio. La figura siguiente muestra el número de patentes relacionadas con este segundo grupo.



Como conclusión de hacia dónde puede enfocarse la tecnología NFC, sirva el

siguiente diagrama proporcionado por "NFC Forum"[32], donde se puede observar que la tecnología NFC puede usarse en cualquier ámbito donde se produzca una comunicación, ya sea activa (intercambio de acciones) o pasiva (lectura de etiquetas).

32 *http://members.nfc-forum.org/aboutnfc/nfc_in_action/*

DNI 3.0

Una aplicación especial de nuestro país, con respecto a la tecnología NFC, es el DNI electrónico. Desde el año 2006 todos los Documentos Nacionales de Identidad que se expiden en España son documentos electrónicos, coexistiendo actualmente dos versiones: **DNIe y DNIe 3.0**[33].

El **primer DNIe**, expedido desde 2006 hasta finales de 2015 incorporaba, a diferencia de su predecesor, un chip en el anverso con información digital, y se presentaba por primera vez en una tarjeta de policarbonato con las mismas medidas que una tarjeta de crédito convencional.

Los avances tecnológicos recogidos en el chip permiten al ciudadano conectar con la Administración de forma digital, hacer uso de su identidad electrónica, firmar documentos digitalmente y de forma remota, etc.

Para ello, el ciudadano ha de disponer de un dispositivo hardware que permita la lectura de los certificados contenidos en el chip, y posibilite la conexión a los distintos servicios digitales, así como instalar los diferentes drivers necesarios para el funcionamiento del hardware. Todos los elementos necesarios pueden ser consultados en el portal del DNIe[34].

Los cambios en la sociedad de la información requieren la constante actualización del Documento Nacional de Identidad. Conscientes de la necesidad no sólo de implementar un instrumento de elevada seguridad, sino también de mejorar y acercar a los ciudadanos su usabilidad, la Dirección General de la Policía lanza en 2015 el DNIe 3.0.

33 *Informacion extraida de "http://www.dnielectronico.es/PortalDNIe/PRF1_Cons02.action?pag=REF_038"*
34 *http://www.dnielectronico.es/PortalDNIe/PRF1_Cons02.action?pag=REF_300*

La principal novedad frente a su antecesor, es la presencia de un chip con un interfaz dual, el cual permite la conexión mediante hardware (chip), pero también de forma inalámbrica a través de la tecnología NFC.

Para utilizar la funcionalidad inalámbrica del DNIe 3.0 únicamente será necesario disponer de:

- Un teléfono *Smartphone* o *tablet* con tecnología NFC.
- App del servicio al que nos queremos conectar.

El ciudadano no tendrá por tanto, que descargarse ningún certificado o driver, sino que la conexión se iniciará simplemente con acercar el DNIe 3.0 a la antena NFC del dispositivo, (a una distancia no superior a 1 cm).

NFC y Android

Android ha supuesto un antes y un después en el desarrollo de la tecnología de los dispositivos móviles. Su implacable versatilidad le ha dado, desde su presentación oficial en 2007, una cuota de mercado de más del 80% en 2013 y de cerca del 94% en la actualidad.

Aunque la mayoría de las aplicaciones estén desarrolladas en Java, Android no tiene una máquina Java, sino que el código original se optimiza para correr sobre una máquina virtual, ya sea Dalvik o Art[35], de manera mucho más eficiente.

Es por esto que Android ha desarrollado un soporte propio que puede verse completo en "Android Developres"[36], y que se resume a continuación.

Dando permisos de acceso

Como pasa con todas las aplicaciones, Android necesita tener los permisos para acceder a los recursos con los que va a trabajar y saber si éstos existen. Hay que tener en cuenta que la versión mínima del API es la 10, mientras que la versión 14 posee una manera más fácil de enviar *NDEF messages* (la versión 9 proporciona un acceso limitado mediante ACTION_TAG_DISCOVERED). En este sentido, la implementación para conceder acceso a NFC sería:

```
<uses-permission android:name="android.permission.NFC" />

<uses-feature android:name="android.hardware.nfc" android:required="true" />

<uses-sdk android:minSdkVersion="10"/>
```

Si la aplicación usa NFC pero la funcionalidad no es crítica para la aplicación, entonces puede omitirse la declaración del `uses-feature` element y comprobar en tiempo de ejecución si está mediante "`getDefaultAdaptar() is null`"

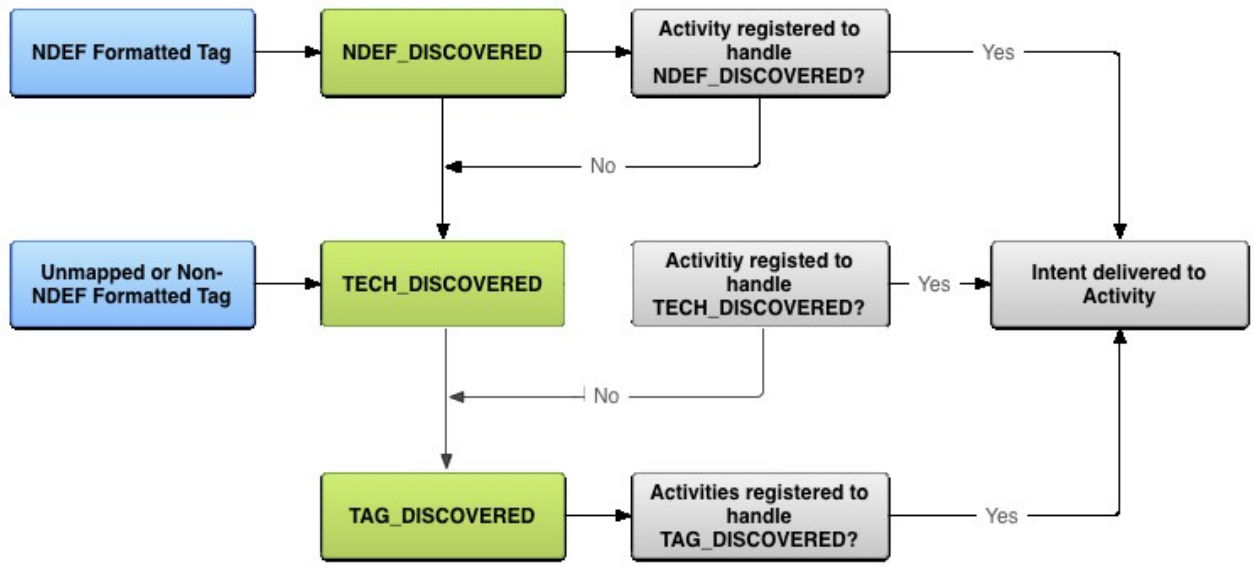
Para filtrar el descubrimiento de un mensaje se utilizan los `Intents`. Recordamos que un `Intent` es un mecanismo para invocar aplicaciones externas a la nuestra,

35 *http://www.elandroidelibre.com/2013/11/art-la-nueva-maquina-virtual-de-google-que-sustituira-a-dalvik.html*
36 *http://developer.android.com/guide/topics/connectivity/nfc*

lanzar eventos a los que otras aplicaciones puedan responder, etc. Debemos por lo tanto registrar dicha actividad en el manifiesto. A continuación se muestra un ejemplo de filtro para la URI "http://developer.android.com/index.html".

```
<intent-filter>
    <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:scheme="http"
      android:host="developer.android.com"
       android:pathPrefix="/index.html" />
</intent-filter>
```

El sistema de reconocimiento de etiquetas funciona según el siguiente esquema.



Cuando se trabaja con etiquetas NFC y dispositivos con Android, el formato principal que se utiliza para leer y escribir datos en las etiquetas es NDEF. Cuando un dispositivo escanea una etiqueta con los datos NDEF, Android proporciona soporte al analizar el mensaje y entregarlo en un `NdefMessage` cuando sea posible.

En los casos en los que al escanear una etiqueta, ésta no contenga datos NDEF o los datos NDEF no pueden ser comparados con un tipo MIME o URI, entonces es necesario abrir la comunicación directa con la etiqueta para leer y escribir en ella con su propio protocolo (en bytes sin formato). Android proporciona soporte genérico para estos casos de uso con el paquete `android.nfc.tech`, que se describe en la Tabla que se muestra a continuación. Se puede utilizar el

`getTechList()` para determinar las tecnologías soportadas por la etiqueta y crear el objeto `TagTechnology` correspondiente a una de las clases proporcionadas por `android.nfc.tech`

Class	Description
TagTechnology	The interface that all tag technology classes must implement.
NfcA	Provides access to NFC-A (ISO 14443-3A) properties and I/O operations.
NfcB	Provides access to NFC-B (ISO 14443-3B) properties and I/O operations.
NfcF	Provides access to NFC-F (JIS 6319-4) properties and I/O operations.
NfcV	Provides access to NFC-V (ISO 15693) properties and I/O operations.
IsoDep	Provides access to ISO-DEP (ISO 14443-4) properties and I/O operations.
Ndef	Provides access to NDEF data and operations on NFC tags that have been formatted as NDEF.
NdefFormatable	Provides a format operations for tags that may be NDEF formattable.
MifareClassic	Provides access to MIFARE Classic properties and I/O operations, if this Android device supports MIFARE.
MifareUltralight	Provides access to MIFARE Ultralight properties and I/O operations, if this Android device supports MIFARE.

Por ejemplo, para obtener una instancia de una etiqueta deberemos hacer un `intent`.

```
Tag tagFromIntent = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
```

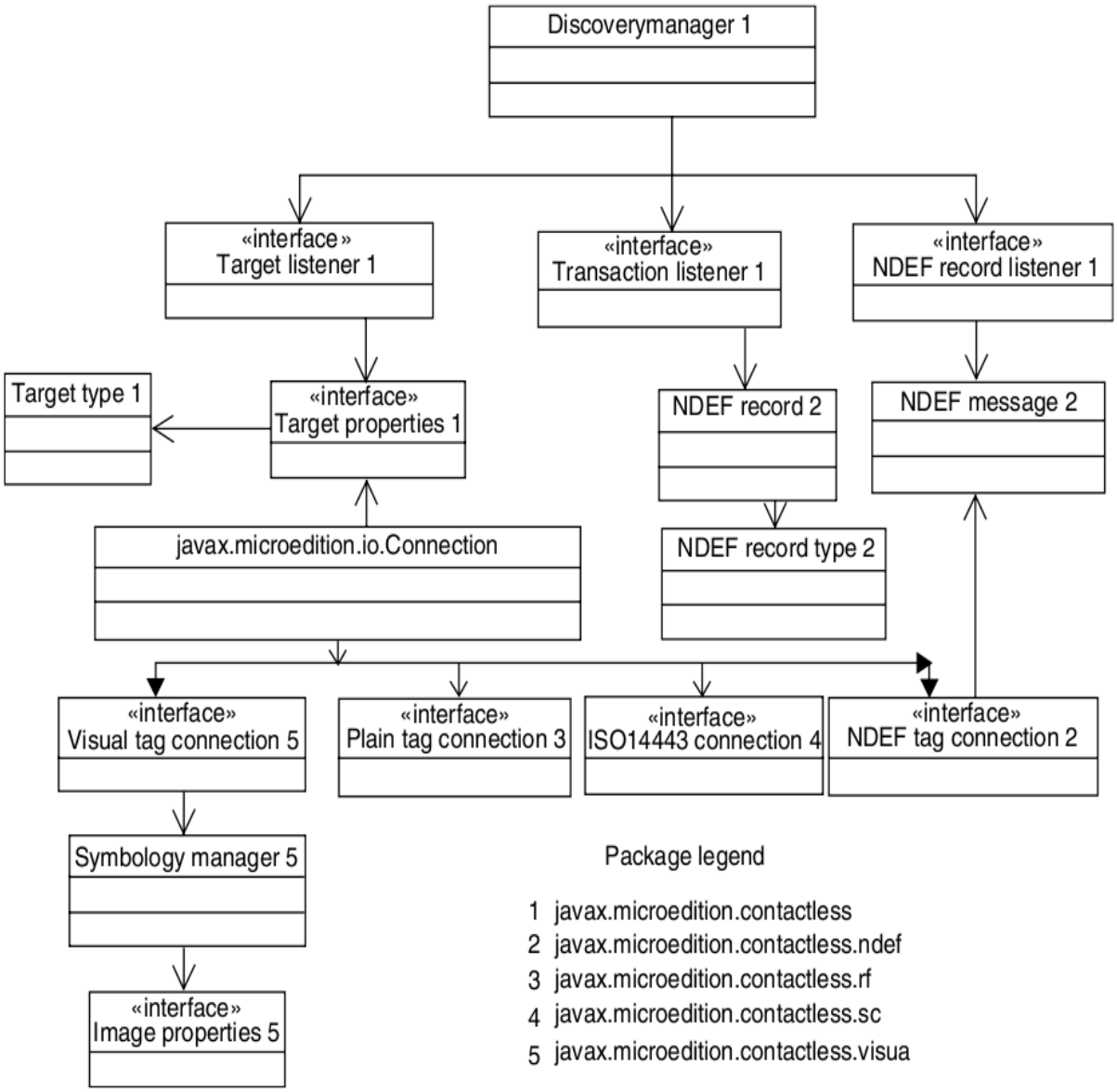
Más adelante se presentará un ejemplo sencillo de manejo de etiquetas en Android.

pagina en blanco
intencionada

NFC y Java

NOKIA proporciona el API JSR-257 para JavaME, el cual se denomina *Contactless Communication API,* y que nos permite poder manejar NFC desde Java..

Los componentes de la API JSR-257 son:



Cabe destacar que de los tres modos de funcionamiento, emulación de NFC-

Card, Lectura/Escritura y punto a punto (P2P), este último es el único no soportado por el API. A continuación, se presentan algunos ejemplos de utilización de dicha API.

Descubriendo y escuchando objetivos soportados

Utilizamos para ello el método

`DiscoveryManager.getSupportedTargetTypes()`

el cual nos devuelve un array de `TargetTypes`, para después utilizar

`registerTargetListener()`

para registrar dichos dispositivos.

```
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) {
        // ...
    }
}
```

La API define los siguiente tipos de objetivos: `ISO14443_CARD`, `NDEF_TAG`, `RFID_TAG` and `VISUAL_TAG`.

Una vez descubiertos y registrados los objetivos podemos obtener las propiedades de los mismos.

```
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) {
            // ...
        }
    }
}
```

Este método realiza las siguientes acciones:

- Recibe las etiquetas de propiedades (`TargetProperties`) de los objetivos detectados,
  - Para cada objetivo se obtiene su URL.
  - Se realiza la conexión usando GCF (Generic Connection Framework).
  - Se realiza el intercambio de datos.
  - Los mensajes entrantes son procesados según sus atributos.

– Finaliza liberando los recursos y cerrando las conexiones

Escuchando objetivos específicos

La API provee de métodos para descubrir unos determinados objetivos. Para ello no necesita detalles, todo lo que necesita es conocer el tipo de mensaje y cómo procesarlo.

Para realizar estas acciones se utiliza la interfaz `NDEFRecordListener` y su método `recordDetected(NDEFMessage ndefMessage)` para registrarlo después con `addNDEFRecordListener(listener, recordType)`.

```
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) {
}
```

Los tipos definidos para un registro NDEF son:

- **EMPTY**- el registro se identifica como vacío.
- **EXTERNAL_RTD** – el registro se identifica como una aplicación específica que sigue las convenciones de nombre de NFC.
- **MIME** – es de tipo MIME según se define el la RFC 2046.
- **NFC_FORUM_RTD** – identifica un tipo de registro "NFC Forum".
- **UNKNOWN** – para identificar un tipo de registro desconocido.
- **URI** – identificador para URIs tal y como se define en la RFC 3986.

Procesando mensajes NDEF

```
/**
 * Called by the platform, when the requested NDEF record type is
 * discovered by the device from the contactless target.
 *
 * @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
        // ...
    }
}
```

El tratamiento de los datos del mensaje depende de la aplicación. Por ejemplo después de recibir la URL, ésta puede abrirse en un navegador.

Registrando la actividad en emulación de tarjeta

```
import javax.microedition.contactless.TransactionListener;
// Register Transaction Listener

try {
    dm.addTransactionListener(this);
} catch (IllegalStateException e) {
    ...
```

```
} catch (Exception e) {
    ...
}

/**
 * Called by the platform, when a card emulation event
 * has happened on the RFID hardware.
 *
 * @param slot is the slot needed to open the APDUConnection defined
 *             in JSR 177 to the external secure element, may be
 *             UNKNOWN_SLOT constant defined in this interface, if the
 *             slot can not be identified.
 */

public void externalReaderDetected(byte slot) {

    // Based on slot number above, using ISO14443Connection or SATSA
    // connect to applet, query applet, update screen, etc.
    ...


}
```

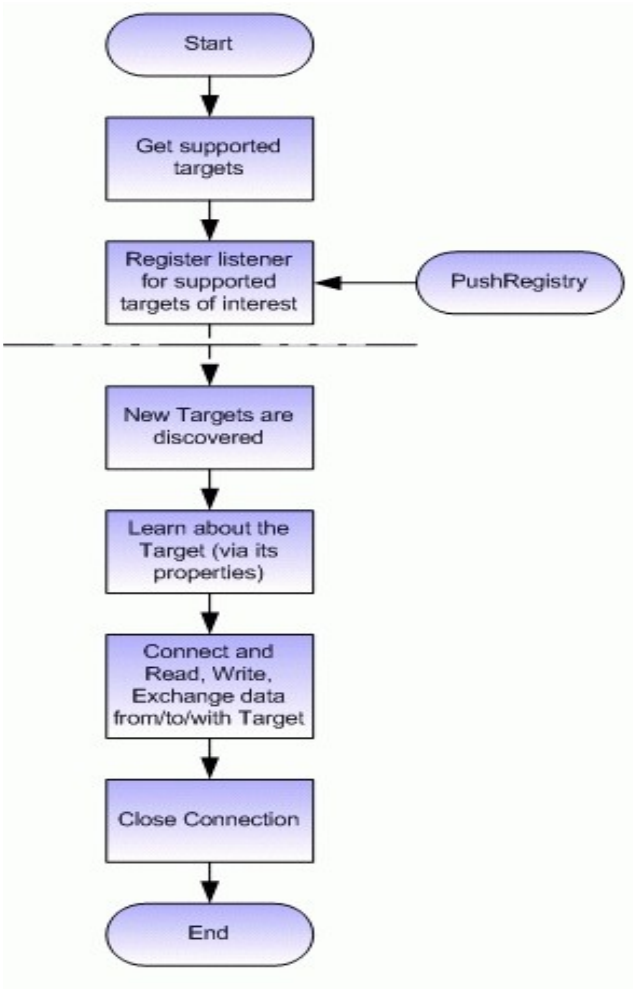
Usando PushRegistry para lanzar aplicaciones NFC.

En las aplicaciones NFC, la activación automática al iniciar es muy importante para el arranque del MIDlets[37], ya que dicha aplicación debe iniciarse en cuanto se acerca el dispositivo.

PushRegistry facilita esta parte en el dispositivo móvil (MIDP[38]), lo que permite la ejecución automática de aplicaciones en tiempo de conexión. Hay que advertir que la API extiende *PushRegistry* para la aplicación solo para los registros NDEF y la actividad en elementos seguros (emulación de tarjeta).

37 *Programa desarrollado con el lenguaje de programación Java para dispositivos embebidos*
38 *http://www.oracle.com/technetwork/systems/index-156665.html (The MIDP 2.0 Push Redistry)*

En la siguiente figura puede verse como interactúa *PushRegistry* para desencadenar el proceso de descubrimiento de nuevos objetivos.



El formato de URL para NDEF que permite realizar el lanzamiento de aplicaciones es:

"ndef:"<record_type_format>?name=<record_type_string>

donde:

– <record_type_format> es "rtd", "external_rtd", "mime", or "uri",

– <record_type_string> es una cadena UTF-8 que identifica el nombre completo según lo especificado por la aplicación:

por ejemplo, urn:nfc:wkt:Sp (Nokia Smart-posters)

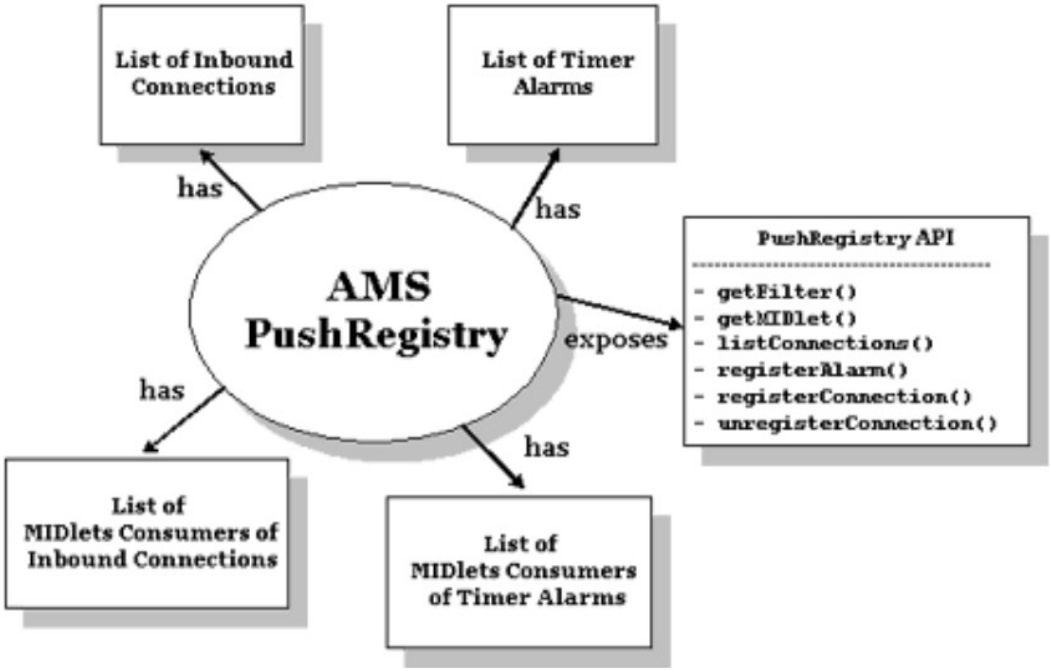
El formato de URL para Emulación de Tarjeta es:

`"secure-element:"?aid=`<aid_string>

donde:

<aid_string> es una cadena de números que contiene el identificador del applet especificado (por ISO7816-5)

Para asegurarse que la aplicación registre tan pronto como sea posible los registros NDEF, se ejecutarán los métodos `NDEFRecordListener` y `TransactionListener`. La siguiente figura resume los elementos del *PushRegistry*.



NFC y Python

Python[39] es un lenguaje interpretado basado en scripts. Es por lo tanto multiplataforma, de propósito general y orientado a objetos. Aunque su expansión y popularidad es reciente, fue creado en 1989 por Guido van Rossum.

Al contrario que otros lenguajes, basa su potencial en la extensa colección de librerías disponibles y en la facilidad de importación y uso de las mismas. Además no necesita definir el tipo de variable y tiene una gran potencia en el manejo de listas y arrays.

Los programas en python pueden ejecutarse con `python nombreprograma.py`, y desde dentro de un entorno pyhton haciendo `run nombreprograma.py` o línea a línea.

Para este último caso bastará con abrir un terminal y ejecutar python, lo que nos devolverá la versión de python, y el cursor del entorno del mismo "`>>>`". A partir de ese momento ya podemos ejecutar sentencias.

```
krilin@namec:~$ python3.4
Python 3.4.3 (default, Sep 14 2016, 12:36:27)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> ... a escribir en python
```

Una vez dentro del entorno, la librería que permite acceder a NFC es `nfcpy`[40]. Para incorporarla a nuestro código bastaría con la siguiente sentencia:

```
import nfc
```

A continuación se presentan diversos ejemplos de su uso para diversas funcionalidades.

39 *https://www.python.org/*
40 *https://nfcpy.readthedocs.io/en/latest/*

Enviando un Link al teléfono

```
>>> import nfc, nfc.snep, threading
>>> connected = lambda llc: threading.Thread(target=llc.run).start()
>>> uri = nfc.ndef.Message(nfc.ndef.UriRecord("http://nfcpy.org"))
>>> clf = nfc.ContactlessFrontend('usb')
>>> llc = clf.connect(llcp={'on-connect': connected})
>>> nfc.snep.SnepClient(llc).put(uri)
True
>>> clf.close()
```

Construyendo registros NDEF

La clase `nfc.ndef.Message` puede ser inicializada con un mensaje NDEF enmarcado por las marcas de principio y fin en el primer y último mensaje, Cada registro NDEF está representado por un objeto `nfc.ndef.Record` accesible mediante un índice o realizando una iteración sobre dicho objeto.

```
>>> import nfc.ndef
>>> message = nfc.ndef.Message(b'\xD1\x01\x0ET\x02enHello World')
>>> message
nfc.ndef.Message([nfc.ndef.Record('urn:nfc:wkt:T', '', '\x02enHello World')])
>>> len(message)
1
>>> message[0]
nfc.ndef.Record('urn:nfc:wkt:T', '', '\x02enHello World')
>>> for record in message:
>>>     record.type, record.name, record.data
>>>
('urn:nfc:wkt:T', '', '\x02enHello World')
```

Si el contenido es inválido o no contiene datos, al analizar el mensaje NDEF se produce un error `nfc.ndef.FormatError` o `nfc.ndef.LengthError` que puede ser propagado.

```
>>> try: nfc.ndef.Message('\xD0\x01\x00')
... except nfc.ndef.LengthError as e: print e
...
insufficient data to parse
>>> try: nfc.ndef.Message('\xD0\x01\x00T')
... except nfc.ndef.FormatError as e: print e
...
ndef type name format 0 doesn't allow a type string
```

Creando mensajes NDEF

Para construir mensajes NDEF se usa la clase `nfc.ndef.Record` para crear los registros, creando los mensajes con `nfc.ndef.Message` y dichos registros como argumentos.

```
>>> import nfc.ndef
>>> record1 = nfc.ndef.Record("urn:nfc:wkt:T", "id1", "\x02enHello World!")
>>> record2 = nfc.ndef.Record("urn:nfc:wkt:T", "id2", "\x02deHallo Welt!")
>>> message = nfc.ndef.Message(record1, record2)
```

La clase `nfc.ndef.Message` puede aceptar también una lista de registros como un solo argumento. Además, es posible añadir registros mediante `nfc.ndef.Message.append()` o `nfc.ndef.Message.extend()`.

```
>>> message = nfc.ndef.Message()
>>> message.append(record1)
>>> message.extend([record2, record3])
```

La serialización de un mensaje definido en un objeto `nfc.ndef.Message` se realiza con la función `str()`.

```
>>> message = nfc.ndef.Message(record1, record2)
>>> str(message)
'\x99\x01\x0f\x03Tid1\x02enHello World!Y\x01\x0e\x03Tid2\x02deHallo Welt!'
```

Los registros específicos son también muy sencillos de crear.

Registro de tipo Texto

```
>>> import nfc.ndef
>>> record = nfc.ndef.TextRecord("Hello World!")
>>> print record.pretty()
text     = Hello World!
language = en
encoding = UTF-8
```

Registro tipo Uri

```
>>> import nfc.ndef
>>> record = nfc.ndef.UriRecord("http://nfcpy.org")
>>> print record.pretty()
uri = http://nfcpy.org
```

Registro tipo Smart Poster

```
>>> import nfc.ndef
>>> uri = "https://launchpad.net/nfcpy"
>>> record = nfc.ndef.SmartPosterRecord(uri)
>>> record.title = "Python module for near field communication"
>>> record.title['de'] = "Python Modul für Nahfeldkommunikation"
>>> print record.pretty()
```

```
resource  = https://launchpad.net/nfcpy
title[de] = Python Modul für Nahfeldkommunikation
title[en] = Python module for near field communication
action    = default
```

pagina en blanco
intencionada

NFC y Raspberry Pi

Raspberry Pi es un ordenador de bajo coste desarrollado en Reino Unido por la fundación del mismo nombre. Por lo tanto, puede ejecutar una distribución linux que tenga instalado python y de esta manera, podrá trabajar con NFC. Un aspecto a tener en cuenta es que la arquitectura de la Raspberry Pi es ARM (y no i388) por lo que es posible que las librerías como `nfcpy` no funcionen correctamente con algunos lectores.

Este apartado presenta una solución para usar un módulo NFC de la firma NXP[41] especial para la Raspberry Pi. Esta solución no tiene soporte para la librería `nfcpy`, pero sí que posee programas de ejemplo escritos en lenguaje C. Sus características son:

- Basada en el chip NXP PN512 que cumple las especificaciones completas del "NFC Forum", y que funciona en los modos lector, P2P y emulador de tarjeta.
- Posibilidad de usar interfaces SPI o I²C[42], aunque el software actual solo soporta SPI.
- Antena de altas prestaciones integrada.
- Incluye una tarjeta RFID MIFARE Ultralight.

Este módulo NFC está especialmente diseñado para trabajar con la Raspberry Pi. El código puede descargarse de la web de NXP[43]. En la imagen siguiente puede verse dicha placa.

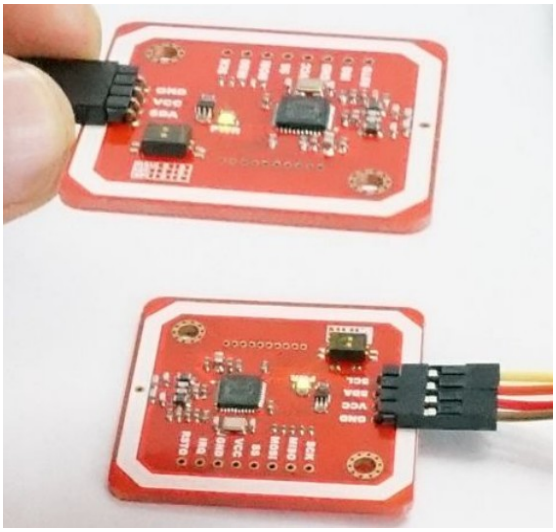
41 *http://www.nxp.com*
42 *Tipos de bus de datos (http://www.i2c-bus.org/)*
43 *http://www.nxp.com/documents/software/SW282711.zip*

NFC y Arduino

Arduino es una compañía de hardware libre y una comunidad tecnológica que diseña y manufactura placas de desarrollo de hardware y software. Estas placas de desarrollo están compuestas respectivamente por circuitos impresos que integran un microcontrolador y un entorno de desarrollo (IDE), en donde se programa cada placa. Arduino se enfoca en acercar y facilitar el uso de la electrónica y la programación de sistemas embebidos en proyectos multidisciplinares. Toda la plataforma, tanto para sus componentes de hardware como de software, es liberada con licencia de código abierto[44], lo que permite libertad de acceso a los distintos componentes.

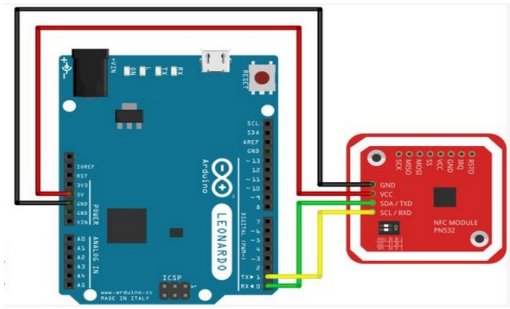
Existen numerosos ejemplos en internet del uso de NFC con Arduino. En esta sección tomaremos como base un sencillo ejemplo extraído de www.instructables.com. Para este ejemplo, se necesita una tarjeta de expansión. Existen varias en el mercado, como pueden ser Adafruit, ElectHouse, SpeedStudio. Todas están basadas en el chip PN532 de NXP, luego su programación y configuración son similares.

La placa de este ejemplo es la PN532 NF RFID module v3[45]



44 *https://es.wikipedia.org/wiki/Código_abierto*
45 *http://www.elechouse.com/elechouse/index.php?main_page=product_info&cPath=90_93&products_id=2242*

La conexión con Arduino se realiza tal y como se muestra en la siguiente figura.



Existen diferentes librerías que nos facilitarán la programación para Arduino. En este ejemplo se han utilizando la de Electhouse y Don Coleman que se puede descargar desde Github. La primera se utiliza para la comunicación con el chip PN532[46], y la segunda para el protocolo de NFC y Mifare[47].

El siguiente código representa un ejemplo de la lectura de etiquetas, imprimiendo los datos leídos.

```
#include <PN532_HSU.h>
#include <PN532.h>
#include <NfcAdapter.h>
PN532_HSU pn532hsu(Serial1);
NfcAdapter nfc = NfcAdapter(pn532hsu);
void setup(void) {
    Serial.begin(9600);
    Serial.println("NDEF Reader");
    nfc.begin();
}
void loop(void) {
    Serial.println("\nScan a NFC tag\n");
    if (nfc.tagPresent()) {
        NfcTag tag = nfc.read();
        tag.print();
    }
    delay(1000);
}
```

El siguiente ejemplo muestra un sistema de control de presencia.

```
        /************************************************
                        Autor: Gonzalo Matarrubia
```

46 *https://github.com/elechouse/PN532*
47 *https://github.com/don/NDEFEsta*

```
            Proyecto: 1er tutorial NFC para Geeky Theory
          ************************************************
          Utilizar el número de identificación único de cada
          etiqueta NFC para "fichar" una entrada o salida
          ************************************************/
//Incluimos la librería de la shield NFC
#include <PN532.h>

//Macros
#define SCK         13
#define MOSI        11
#define SS          10
#define MISO        12

#define miTarjeta 251112222
#define fuera 0
#define dentro 1

//Creamos un objeto NFC de clase PN532
PN532 NFC(SCK,MISO,MOSI,SS);

//Creamos variables globales
int persona = fuera;
void setup () {
     Serial.begin(9600);
     if(!Serial) { delay(100); }

     //Configuramos el NCF
     NFC.begin();
     NFC.SAMConfig();
}


void loop () {
     /* Leemos constantemente el número de identificación mientras no haya
     ninguna etiqueta NFC que leer id valdrá 0 */
     uint32_t id = NFC.readPassiveTargetID(PN532_MIFARE_ISO14443A);

     if( miTarjeta == id) {
          //Comprobamos si la persona estaba fuera o dentro
          if ( persona == fuera ){
               Serial.println("Bienvenido");
               persona=dentro;
               delay(2000);
```

```
          }
          else {
               Serial.println("Hasta luego");
               persona=fuera;
               delay(2000);
          }
     }
     delay(50);
}
```

Accediendo a una etiqueta NFC

Por último, un ejemplo de cómo cambiar la clave[48] de acceso a los datos de una etiqueta.

Partimos de la base de que tenemos una *shield*[49] de Arduino con la que podemos leer o escribir datos de la memoria EEPROM de una etiqueta, en este caso una Mifare 1k. La clave para autenticarnos y poder acceder a la información la sabemos porque usamos la que traía nuestro llavero por defecto, y en general la que suelen traer todas las etiquetas Mifare 1k.

A continuación vamos a ver cómo cambiar la clave que viene por defecto y cómo hacer que nuestra clave no se pueda cambiar más o que la información que contiene no se pueda borrar, entre otras cosas. Cabe destacar que podemos cambiar la clave o los permisos de escritura/lectura por sectores, es decir, que podemos tener un sector con información oculta pero dejar accesible el resto.

El siguiente código realiza un cambio de la KEY_A del sector 1. No se cambia ninguno de los permisos.

```
     /*  ************************************************
                    Autor: Gonzalo Matarrubia
                  tutorial NFC para Geeky Theory
          ************************************************
          Descripción: Este sketch hace una lectura
           del sector 1. Tras la primera lectura hace
           un cambio de KEY_A en el mismo sector.
```

48 *https://geekytheory.com/nfc-arduino-parte-3/*
49 *Extensiones para la placa Arduino*

```
               Luego sigue haciendo lecturas y no
            pasaremos el proceso de autentificación.
          ************************************************/
//Incluimos la librería de la shield NFC
#include <PN532.h>

//Macros
#define SCK         13
#define MOSI        11
#define SS            10
#define MISO        12

//Creamos un objeto NFC de clase PN532
PN532 NFC(SCK,MISO,MOSI,SS);
void setup () {
     Serial.begin(9600);
     if(!Serial) { delay(100); }

     //Configuramos el NCF
     NFC.begin();

     //Comprobamos que tenemos puesta la shield
     uint32_t versiondata = NFC.getFirmwareVersion();
     if( ! versiondata ) {
          while (1) { delay (100); } //No continuamos con el programa
     }
     //Configuramos el NFC para la lectura y escritura
     NFC.SAMConfig();
}
void loop () {
// Leemos el número de identificación, mientras no haya ninguna etiqueta
valdrá 0
     uint32_t id = NFC.readPassiveTargetID(PN532_MIFARE_ISO14443A);

     uint8_t key [ ] = { 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF} ;  //Clave por defecto
     uint8_t key1 [ ] = { 0xAA,0xBB,0xCC,0xDD,0xEE,0xFF} ;  //Nueva Clave

     if ( id ) {
          Serial.print("Encontrada etiqueta #");
          Serial.println(id);

          /* Nos identificamos para el bloque 4 -> el bloque 4 pertenece
             sector 1 que va del bloque 4 al 7. El 7 es el sector trailer. */
          if( NFC.authenticateBlock (1, id, 4, KEY_A, key) ) {
```

```
            //Creamos un vector para almacenar la lectura
            uint8_t lectura[16];
            for(int bloque = 4; bloque<=7; bloque++) {
                 //Leemos los bloques del sector
                if (NFC.readMemoryBlock (1, bloque, lectura) ) {
                      //Mostramos el bloque 4
                      for( int i = 0; i<16 ; i++){
                           Serial.print(lectura[i],HEX);
                           //Para que quede más ordenado
                          Serial.print("  ");
                      }
                      Serial.print (" | Block ");
                      Serial.println(bloque);
                }
                //else Serial.print("Lectura fallida \n");
                //Util para hacer debug
            }
//Creamos un bloque de información para cambiar la clave

        uint8_t datos[16] = {   /*Nueva KEY_A*/ // /*Nueva KEY_A*/
                                 0xAA, 0xBB,  0xCC,  0xDD,  0xEE,  0xFF,
                                 /*Dejamos los mismos permisos*/
                                0xFF, 0x7,  0x80, 0x69,
                                 /*Dejamos la misma KEY_B*/
                                0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

        Serial.print("Cambiando clave... \n");
        NFC.writeMemoryBlock (1,7, datos);
        Serial.print("Clave cambiada. \n");
       }
  else Serial.print("Autentificacion fallida \n"); //Util para hacer debug
  }
  delay(1500);}
```

Ejemplo Android: Editor de etiquetas

Se presenta ahora un sencillo ejemplo de programación de etiquetas con Android.

Lo primero es adecuar el "AndroidMinifest.xml".

```
<?xml version="1.0" encoding="utf-8"?>
      <manifest xmlns:android="http://schemas.android.com/apk/res/android"
            package="com.androcode.nfc"
            android:versionCode="1"
            android:versionName="1.0" >
      <uses-sdk
            android:minSdkVersion="8"
            android:targetSdkVersion="17" />
      <application
            android:allowBackup="true"
            android:icon="@drawable/ic_launcher"
            android:label="@string/app_name"
            android:theme="@style/AppTheme" >
      <activity
            android:name="com.androcode.nfc.MainActivity"
            android:label="@string/app_name" >
      <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
      </activity>
      </application>
      <uses-permission android:name="android.permission.NFC" />
</manifest>
```

Ahora crearemos el interfaz con un `TextView` y un `EditText`. Podemos utilizar alguno de los editores gráficos disponibles de forma gráfica o programar directamente el layout.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
```

```
    android:orientation="vertical" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Write you message: ">
    </TextView>
    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="20sp" >
        <EditText
            android:id="@+id/edit_message"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:hint="message" />
        <Button
            android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Write!!" />
    </LinearLayout>
</LinearLayout>
```

Ahora se crea la *Activity*.

Como *adapter* usamos el **NfcAdapter** que viene por defecto en nuestro dispositivo con NFC.

El *PendingIntent* que usamos recibirá, como *Flag* el Intent.FLAG_ACTIVITY_SINGLE_TOP, de tal manera que la *Activity* no creará uno de nuevo y lo reutilizará pues ya está presente en la pila de actividades.

Luego definimos el *IntentFilter*. Este *IntentFilter* será el que nos indique si se ha encontrado un *Tag* (etiqueta). Y a nuestro array de *IntentFilters* le añadimos éste que hemos creado.

El siguiente paso es enviar el mensaje a escribir y el *Tag* al método *write.* Y por último, en *createRecord* creamos el *NdefRecord*. Con el objeto *Ndef* obtenemos el *Tag*, escribimos el mensaje *NdefRecord* y cerramos.

Comprendiendo los mensajes NFC

Un mensaje NDEF está representado por la clase *NdefMessage*, que está compuesto por varios registros encapsulados de tipo *NdefRecord*. El primer registro de un mensaje NDEF determina cómo interpretar el conjunto del mensaje. Un registro *NdefRecord* tiene los siguientes campos:

- **3-bit TNF** (*Type Name Format*) Campo que indicará cómo se interpretará el mensaje.
- **length type**: Formato del registro.
- **length ID:** Identificador único del registro.
- **length payload:** Datos.

Tipos de formato. Los TNFs vienen definidos por los siguientes tipos:

TNF_EMPTY (Mensaje vacío)

TNF_WELL_KNOWN (Mensaje bien definido.)

TNF_MIME_MEDIA (Registro definido con un MIME-type)

TNF_ABSOLUTE_URI (URI), **TNF_EXTERNAL_TYPE** (Tipo especial)

TNF_UNKNOWN (Desconocido)

TNF_UNCHANGED (Trozo del mensaje)

Cuando el TNF es del tipo TNF_WELL_KNOWN significa que el mensaje NDEF tiene alguno de los siguientes RTDs: **RTD_TEXT** (Texto plano con ISO del idioma correspondiente), **RTD_URI** (Contiene una URI), **RTD_SMART_POSTER** (Contiene varios registros como URI, acciones, etc.)

A continuación, se muestra el código que implementa el acceso a NFC con Android.

```
package com.androcode.nfc;

import java.io.IOException;
import java.io.UnsupportedEncodingException;

import android.nfc.FormatException;
import android.nfc.NdefMessage;
import android.nfc.NdefRecord;
```

```
import android.nfc.NfcAdapter;
import android.nfc.Tag;
import android.nfc.tech.Ndef;
import android.os.Bundle;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

@SuppressLint("NewApi")
public class MainActivity extends Activity {
    NfcAdapter adapter;
    PendingIntent pendingIntent;
    IntentFilter writeTagFilters[];
    boolean writeMode;
    Tag myTag;
    Context context;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //Definimos el layout a usar
        setContentView(R.layout.activity_main);
        context = this;
        //Los elementos que vamos a usar en el layout
        Button btnWrite = (Button)findViewById(R.id.button);
        final TextView message =
(TextView)findViewById(R.id.edit_message);
        //setOnCLickListener hará la acción que necesitamos
        btnWrite.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
      try{
            //Si no existe tag al que escribir, mostramos un mensaje de
```

```
          que no existe.
             if(myTag == null){
       Toast.makeText(context, context.getString(R.string.error_notag),
          Toast.LENGTH_LONG).show();
         }else{
              //Llamamos al método write que definimos más adelante donde
                le pasamos por
              //parámetro el tag que hemos detectado y el mensaje a
                escribir.
       write(message.getText().toString(),myTag);
           Toast.makeText(context, context.getString(R.string.ok_write),
          Toast.LENGTH_LONG).show();
        }
      }catch(IOException e){
  Toast.makeText(context,
    context.getString(R.string.error_write),Toast.LENGTH_LONG).show();
      e.printStackTrace();
  }catch(FormatException e){
     Toast.makeText(context, context.getString(R.string.error_write),
    Toast.LENGTH_LONG).show();
  e.printStackTrace();
  }
   }
});


      adapter = NfcAdapter.getDefaultAdapter(this);
      pendingIntent = PendingIntent.getActivity(this, 0, new
    Intent(this,getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP),
          0);
      IntentFilter tagDetected = new
    IntentFilter(NfcAdapter.ACTION_TAG_DISCOVERED);
      tagDetected.addCategory(Intent.CATEGORY_DEFAULT);
      writeTagFilters = new IntentFilter[]{tagDetected};
  }
  //El método write es el más importante, será el que se encargue de
              crear el mensaje
  //y escribirlo en nuestro tag.
  private void write(String text, Tag tag) throws IOException,
    FormatException{
    //Creamos un array de elementos NdefRecord. Este Objeto representa
          un registro del mensaje NDEF
    //Para crear el objeto NdefRecord usamos el método
    createRecord(String s)
    NdefRecord[] records = {createRecord(text)};
    //NdefMessage encapsula un mensaje Ndef(NFC Data Exchange Format).
          Estos mensajes están
```

```
      //compuestos por varios registros encapsulados por la clase
                NdefRecord
      NdefMessage message = new NdefMessage(records);
      //Obtenemos una instancia de Ndef del Tag
      Ndef ndef = Ndef.get(tag);
      ndef.connect();
      ndef.writeNdefMessage(message);
      ndef.close();
    }
    //Método createRecord será el que nos codifique el mensaje para crear
un NdefRecord
    @SuppressLint("NewApi") private NdefRecord createRecord(String text)
throws UnsupportedEncodingException{
        String lang = "us";
        byte[] textBytes = text.getBytes();
        byte[] langBytes = lang.getBytes("US-ASCII");
        int langLength = langBytes.length;
        int textLength = textBytes.length;
        byte[] payLoad = new byte[1 + langLength + textLength];
        payLoad[0] = (byte) langLength;
        System.arraycopy(langBytes, 0, payLoad, 1, langLength);
        System.arraycopy(textBytes, 0, payLoad, 1+langLength, textLength);
        NdefRecord recordNFC = new NdefRecord(NdefRecord.TNF_WELL_KNOWN,
                              NdefRecord.RTD_TEXT, new byte[0], payLoad);
        return recordNFC;
    }
    //en onnewIntent manejamos el intent para encontrar el Tag
    @SuppressLint("NewApi") protected void onNewIntent(Intent intent){
        if(NfcAdapter.ACTION_TAG_DISCOVERED.equals(intent.getAction())){
            myTag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
            Toast.makeText(this, this.getString(R.string.ok_detected) +
                 myTag.toString(), Toast.LENGTH_LONG).show();
        }
    }
     public void onPause(){
        super.onPause();
        WriteModeOff();
    }
    public void onResume(){
        super.onResume();
        WriteModeOn();
    }
    @SuppressLint("NewApi") private void WriteModeOn(){
        writeMode = true;
```

```
        adapter.enableForegroundDispatch(this,pendingIntent,
                writeTagFilters,null);
    }
     @SuppressLint("NewApi") private void WriteModeOff(){
        writeMode = false;
        adapter.disableForegroundDispatch(this);
    }
}
```

Open NFC

Open NFC es un conjunto de aplicaciones de código abierto que implementan las funcionalidades NFC para diferentes sistemas operativos. De hecho ofrece un conjunto completo de API NFC para Android, Linux y Windows Mobile.

Las principales características de Open NFC son:

- Soporta una gran variedad de etiquetas NFC incluyendo los protocolos de alto nivel y los interfaces de bajo nivel.
- Soporta los diferentes modos de operación: Lectura, Emulación de Tarjeta y Punto-a-Punto.
- Permite utilizar NFC para realizar emparejamiento WIFI y *Bluetooh*.
- Soporta la emulación de elementos seguros e integra una pila de seguridad para filtrar el acceso
- Proporciona NFC *Hardware-independent stack*, basándose en la abstracción de hardware. Se contemplan módulos de NFC HAL para

muchos integrados NFC, incluyendo un simulador

- La pila se proporciona en tres arquitecturas diferentes que le permiten adaptarse a los diferentes entornos: *monolithic*[50], usuario o cliente-servidor.

Las ediciones disponibles en este proyecto de código abierto son:

- SDK (Edición PC).
- Core.
- Linux.
- Android.

Dada la velocidad de los cambios en la tecnología de los dispositivos sobre los que puede desarrollarse NFC, ya no van a tener continuidad:

- J-Edition (JSR257).
- Windows CE Edition.
- BlackBerry.

En las siguientes tablas se resumen las tarjetas, el protocolo utilizado, quien las fabrica y cuáles están soportadas por Open NFC.

50 *En la ingeniería de software, una aplicación monolítica describe una única aplicación en los que la interfaz de usuario, la lógica y el código de acceso a datos se combinan en un solo programa de una plataforma única.*

Card name	Manufacturer	NFC Protocol	Support in MicroRead / Open NFC version	Id.	R/W	Format.	Card Rem.	Comments
NFC Forum Type 1 tags	Any	ISO 14443-A, no anti-collision.	All versions	Y	Y	n/a (see specific cards models for support)	Y	Memory structure and management, RF interface, Transmission handling, Command set, NDEF detection and access as described in NFC Forum specification [link] are fully supported with these tags. This means for example that a locked block in a tag can never be unlocked or written.
NFC Forum Type 2 tags		ISO 14443-A, with anti-collision.		Y	Y		Y	
NFC Forum Type 3 tags		JIS X 6319-4		Y	Y		Y	
NFC Forum Type 4 tags		ISO 14443-A or -B, with anti-collision. APDU based.		Y	Y		Y	
Generic ISO 15693 tags (RFID) / Android “NfcV” tags	Any	ISO 15693.	Open NFC 4.3.1	Y	Y	n/a	Y (*)	While not an NFC Forum specification, this “NfcV” format is supported in Open NFC. This format is similar to but different from INSIDE Secure’s “type 6″ format, also supported. (*) Some ISO 15693-compatible cards do not support card removal detection.
Generic ISO 14443-4 card (type A or B), e.g. MicroPass™ (INSIDE) or MIFARE Smart MX™ (NXP)	Any	ISO 14443-A or -B part 4	All versions	Y	Y	N (*)	Y	Communication is possible using ISO 14443-4 API or ISO 7816-4 helpers. (*) It is possible to write an application using Open NFC to format the card into Type 4 tag. This operation is not generic however.
Generic B Prime cards	Any	B Prime (similar to ISO 14443-B)	All versions	Y	Y	n/a	N	
Kovio NFC Barcode™	Kovio	ISO 14443-A part 2	Open NFC 4.3.0, firmware 7.13	Y	n/a	n/a	N	Based on printed silicon. Tag data (128 bits) is retrieved as a UID through Open NFC.
Kovio 2K	Kovio	ISO 14443-A part 3	Open NFC 4.4.2, firmware 7.13	Y	Y	Y	Y	This tag uses OTP memory, it can be formatted / written only once, in one operation.

Topaz™, Topaz512™, Jewel™	Broadcom (previously Innovision)	ISO 14443-A, no anti-collision.	All versions	Y	Y	Y (except Jewel™)	Y	NDEF and non-NDEF cards are supported in Open NFC.
my-d™ move, my-d™ NFC	Infineon	ISO 14443-3 Type A or ISO 18092.	Open NFC 4.3.0	Y	Y	Y	Y	Password for authentication is specified using WMyDMoveSetConfiguration(). Communication is done over ISO 14443-3 Type A protocol.
MIFARE Classic™ (1K, 4K, mini)	NXP	ISO 14443-A part 3.	Identified in all versions, r/w access requires Open NFC 4.5.2 or Open NFC 4.4.0 with dedicated HAL.	Y	Y	Y (*)	N	Open NFC is delivered with the publicly available TestKey (Application Note AN1304) which is necessary for reading Mifare Classic tags storing NDEF data (referenced as “type 7″ tags). It will not be possible to format a MIFARE Classic card if the key is different. A different key for authentication can be specified with WMifareClassicAuthenticate() for raw commands exchange. The support for Mifare Classic encryption algorithm (card emulation and reader mode) was developed by using publicly available information.
MIFARE Plus™ (S/X, 2K/4K))	NXP	ISO 14443-A part 4.	All versions	Y	N	N	N	Low-level communication with the card is possible using W14Part3ExchangeRawBits(), however it means the cryptography (Crypto1™ or AES) has to be implemented in the end-user application.
MIFARE Ultralight™, Ultralight™ C, Ultralight™ EV1	NXP	ISO 14443-A part 3	All versions	Y	Y	Y	Y	Key for 3DES authentication can be specified with WMifareULCSetAccessRights(). All cards are identified as generic MIFARE Ultralight™.
MIFARE DESFire™ (D40, EV1 2, 4 and 8KB)	NXP	ISO 14443-A part 4	All versions	Y	Y	N (*)	N (*)	Communication is possible using ISO 7816-4 or ISO 14443-4 protocols. It is possible to write an application using Open NFC to format the card into Type 4 tag.

								(*) Card Removal detection works when the card contains the NDEF Type 4 tag application.
ICODE™ cards	NXP	ISO 15693 part 3	All versions	Y	Y	Y	Y	NDEF mapping with Android NfcV format described in NXP's application note AN11032. These cards are fully supported in Open NFC, including access to card data (AFI, DSFID, …)
Tag-it™ cards	TI	ISO 15693 part 3	All versions	Y	Y	Y	Y	These cards are fully supported in Open NFC, including access to card data (AFI, DSFID, …). NDEF formatting done by Open NFC is based on INSIDE Secure "Type 6″ specification.
LRi1K, LRi2K cards	ST	ISO 15693 part 3	All versions	Y	Y	Y	Y	NDEF formatting done by Open NFC is based on INSIDE Secure "Type 6″ specification. These cards are fully supported in Open NFC, including access to card data (AFI, DSFID, …)
Standard FeliCA™ cards	Sony	ISO 18092	All versions	Y	Y	n/a	Y	Only non-encrypted communications at raw ISO 18092 level are supported with these cards.
FeliCA™ lite (RC-S965) cards	Sony	ISO 18092	All versions	Y	Y	n/a	Y	This chip supports NFC Forum Type 3 command set, fully supported in our solution.

Bibliografía y Documentación

Forum NFC	www.nfc-forum.org
Ecma	www.ecma-international.org
Open NFC	open-nfc.org
Wikipedia:	es.wikipedia.org
Nokia	www.developer.nokia.com
Sun	java.sun.com/developer
Near Field Communication org	nearfieldcommunication.org
Android Developers	developer.android.com
NFC y Arduino	www.instructables.com
Mouser Electronics	eu.mouser.com/rfid-nfc
Bada Developers	developer.bada.com
geekytheory.es	geekytheory.com
NFC access control for the Mifare S50 1K	Amal Graafstra

RFID HANDBOOK	ISBN 1-4200-5499-6
Beginnig NFC	O'Reilly ISBN 978-1-449-37206-4
Oracle	www.oracle.com
Libro Blanco de NFC en el transporte publico	ITS España ISBN 978-84-616-4714-9

Palabras claves

NFC, NFCIP, NFC-SEC, NDEF, ECMA, MIFARE, Felicia, Smart Poster, DNI, Android, Raspberry Pi, Python, Arduino, Java.