NFC Data Exchange Format (NDEF)

Technical Specification

NFC Forum™

NDEF 1.0

NFCForum-TS-NDEF_1.0

2006-07-24

Contents

Figures

Tables

Test Requirements

1 Overview

The International Standard ISO/IEC 18092, Near Field Communication – Interface and Protocol (NFCIP-1), defines an interface and protocol for simple wireless interconnection of closely coupled devices operating at 13.56 MHz.

The NFC Data Exchange Format (NDEF) specification defines a message encapsulation format to exchange information, e.g. between an NFC Forum Device and another NFC Forum Device or an NFC Forum Tag.

NDEF is a lightweight, binary message format that can be used to encapsulate one or more application-defined payloads of arbitrary type and size into a single message construct. Each payload is described by a type, a length, and an optional identifier.

Type identifiers may be URIs, MIME media types, or NFC-specific types. This latter format permits compact identification of well-known types commonly used in NFC Forum applications, or self-allocation of a name space for organizations that wish to use it for their own NFC-specific purposes.

The payload length is an unsigned integer indicating the number of octets in the payload. A compact, short-record layout is provided for very small payloads.

The optional payload identifier enables association of multiple payloads and cross-referencing between them.

NDEF payloads may include nested NDEF messages or chains of linked chunks of length unknown at the time the data is generated.

NDEF is strictly a message format, which provides no concept of a connection or of a logical circuit, nor does it address head-of-line problems.

1.1 Objectives

The NFC Data Exchange Format (NDEF) specification is a common data format for NFC Forum Devices and NFC Forum Tags.

The NFC Data Exchange Format specification defines the NDEF data structure format as well as rules to construct a valid NDEF message as an ordered and unbroken collection of NDEF records. Furthermore, it defines the mechanism for specifying the types of application data encapsulated in NDEF records.

The NDEF specification defines only the data structure format to exchange application or service specific data in an interoperable way, and it does not define any record types in detail—record types are defined in separate specifications.

This NDEF specification assumes a reliable underlying protocol and therefore this specification does not specify the data exchange between two NFC Forum Devices or the data exchange between an NFC Forum Device and an NFC Forum Tag. Readers are encouraged to review the NFCIP-1 transport protocol [ISO/IEC 18092].

An example of the use of NDEF is when two NFC Forum Devices are in proximity, an NDEF message is exchanged over the NFC Forum LLCP protocol. When an NFC Forum Device is in proximity of an NFC Forum Tag, an NDEF message is retrieved from the NFC Forum Tag by means of the NFC Forum Tag protocols. The data format of the NDEF message is the same in these two cases so that an NFC Forum Device may process the NDEF information independent of the type of device or tag with which it is communicating.

Because of the large number of existing message encapsulation formats, record marking protocols, and multiplexing protocols, it is best to be explicit about the design goals of NDEF and, in particular, about what is outside the scope of NDEF.

1.1.1 Design Goals

The design goal of NDEF is to provide an efficient and simple message format that can accommodate the following:

1. Encapsulating arbitrary documents and entities, including encrypted data, XML documents, XML fragments, image data like GIF and JPEG files, etc.
2. Encapsulating documents and entities initially of unknown size. This capability can be used to encapsulate dynamically generated content or very large entities as a series of chunks.
3. Aggregating multiple documents and entities that are logically associated in some manner into a single message. For example, NDEF can be used to encapsulate an NFC-specific message and a set of attachments of standardized types referenced from that NFC-specific message.
4. Compact encapsulation of small payloads should be accommodated without introducing unnecessary complexity to parsers.

To achieve efficiency and simplicity, the mechanisms provided by this specification have been deliberately limited to serve these purposes. NDEF has not been designed as a general message description or document format such as MIME or XML. Instead, NFC applications can take advantage of such formats by encapsulating them in NDEF messages.

1.1.2 Anti-Goals

The following list identifies items outside the scope of NDEF:

1. NDEF does not make any assumptions about the types of payloads that are carried within NDEF messages or about the message exchange patterns implied by such messages.
2. NDEF does not in any way introduce the notion of a connection or a logical circuit (virtual or otherwise).
3. NDEF does not attempt to deal with head-of-line blocking problems that might occur when using stream-oriented protocols like TCP.

1.2 References

[ISO/IEC 18092]	ISO/IEC 18092, "Information Technology- Telecommunications and information exchange between systems- Near Field Communication - Interface and Protocol (NFCIP-1)".
[NFC RTD]	"NFC Record Type Definition (RTD) Specification", NFC Forum, 2006.
[RFC 1700]	Reynolds, J. and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994.
[RFC 1900]	B. Carpenter, Y. Rekhter, "Renumbering Needs Work", RFC 1900, IAB, February 1996.
[RFC 2046]	N. Freed, N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types" RFC 2046, Innosoft, First Virtual, November 1996.

[RFC 2047]	K. Moore, “MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text”, RFC 2047, University of Tennessee, November 1996.
[RFC 2048]	N. Freed, J. Klensin, J. Postel, “Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures”, RFC 2048, Innosoft, MCI, ISI, November 1996.
[RFC 2119]	S. Bradner, “Key words for use in RFCs to Indicate Requirement Levels”, RFC 2119, Harvard University, March 1997. http://www.apps.ietf.org/rfc/rfc2119.html
[RFC 2616]	R. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, T. Berners-Lee, “Hypertext Transfer Protocol -- HTTP/1.1”, RFC 2616, U.C. Irvine, DEC W3C/MIT, DEC, W3C/MIT, W3C/MIT, January 1997.
[RFC 2717]	R. Petke, I. King, “Registration Procedures for URL Scheme Names”, BCP: 35, RFC 2717, UUNET Technologies, Microsoft Corporation, November 1999.
[RFC 2718]	L. Masinter, H. Alvestrand, D. Zigmond, R. Petke, “Guidelines for new URL Schemes”, RFC 2718, Xerox Corporation, Maxware, Pirsenteret, WebTV Networks, Inc., UUNET Technologies, November 1999.
[RFC 2732]	R. Hinden, B. Carpenter, L. Masinter, “Format for Literal IPv6 Addresses in URL's”, RFC 2732, Nokia, IBM, AT&T, December 1999.
[RFC 3023]	M. Murata, S. St. Laurent, D. Kohn, “XML Media Types” RFC 3023, IBM Tokyo Research Laboratory, simonstl.com, Skymoon Ventures, January 2001.
[RFC 3986]	T. Berners-Lee, R. Fielding, L. Masinter, “Uniform Resource Identifiers (URI): Generic Syntax”, RFC 3986, MIT/LCS, U.C. Irvine, Xerox Corporation, January 2005. http://www.apps.ietf.org/rfc/rfc3986.html
[URI SCHEME]	List of Uniform Resource Identifier (URI) schemes registered by IANA is available at:http://www.iana.org/assignments/uri-schemes

1.3 Administration

The NFC Forum Data Exchange Format Specification is an open specification supported by the Near Field Communication Forum, Inc., located at:

401 Edgewater Place, Suite 600
Wakefield, MA, 01880

Tel.: +1 781-876-8955
Fax: +1 781-224-1239

http://www.nfc-forum.org/

The Devices technical working group maintains this specification.

1.4 Special Word Usage

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119.

1.5 Name and Logo Usage

The Near Field Communication Forum’s policy regarding the use of the trademarks *NFC Forum* and the NFC Forum logo is as follows:

- Any company MAY claim compatibility with NFC Forum specifications, whether a member of the NFC Forum or not.
- Permission to use the NFC Forum logos is automatically granted to designated members only as stipulated on the most recent Membership Privileges document, during the period of time for which their membership dues are paid.
- Member’s distributors and sales representatives MAY use the NFC Forum logo in promoting member’s products sold under the name of the member.
- The logo SHALL be printed in black or in color as illustrated on the Logo Page that is available from the NFC Forum at the address above. The aspect ratio of the logo SHALL be maintained, but the size MAY be varied. Nothing MAY be added to or deleted from the logos.
- Since the NFC Forum name is a trademark of the Near Field Communication Forum, the following statement SHALL be included in all published literature and advertising material in which the name or logo appears:

  NFC Forum and the NFC Forum logo are trademarks of the Near Field Communication Forum.

1.6 Intellectual Property

The NFC Data Exchange Format (NDEF) Specification conforms to the Intellectual Property guidelines specified in the NFC Forum's Intellectual Property Right Policy, as approved on November 9, 2004 and outlined in the NFC Forum Rules of Procedures, as approved on December 17, 2004.

1.7 Glossary

NDEF application

The logical, higher-layer application on an NFC Forum Device using NDEF to format information for exchange with other NFC Forum Devices or NFC Forum Tags. Also *user application* or *NDEF user application*.

NDEF message

The basic message construct defined by this specification. An NDEF message contains one or more NDEF records (see section 2.3.1).

NDEF record

An NDEF record contains a payload described by a type, a length, and an optional identifier (see section 2.3.2).

NDEF short record

An NDEF record with the *SR* flag set to 1; the PAYLOAD_LENGTH field in short records is a single octet allowing payloads or chunks of up to 255 bytes to be carried (see section 3.2.4).

NDEF record chunk

An NDEF record that contains a chunk of a payload rather than a full payload (see section 2.3.3). Each record chunk carrying a portion of the chunked payload, except the last record of each chunked payload, has its *CF* flag set to 1.

NDEF payload

The application data carried within an NDEF record.

NDEF chunked payload

Application data that has been partitioned into multiple chunks each carried in a separate NDEF record, where each of these records except the last has the *CF* flag set to 1. This facility can be used to carry dynamically generated content for which the payload size is not known in advance or very large entities that don't fit into a single NDEF record. Chunked payloads are not intended to support multiplexing or streaming of content and such use is deprecated. (See section 2.3.3.)

NDEF payload length

The size of the payload in a single NDEF record indicated as the number of octets (see section 2.4.1).

NDEF payload type

An identifier that indicates the type of the payload. This specification supports URIs [RFC 3986], MIME media type constructs [RFC 2616], as well as an NFC-specific record type as type identifiers (see section 2.4.2).

NDEF payload identifier

An optional URI that can be used to identify a payload (see section 2.4.3).

NDEF generator

An entity or module that encapsulates application-defined payloads within NDEF messages.

NDEF parser

An entity or module that parses NDEF messages and hands off the payloads to an NDEF application.

User Application

See *NDEF Application*.

2 NDEF Mechanisms

This section describes the mechanisms used in NDEF. The specific syntax for these mechanisms is defined in Section 3.

2.1 Introduction

NFC Forum Data Exchange Format is a lightweight binary message format designed to encapsulate one or more application-defined payloads into a single message construct. An NDEF message contains one or more NDEF records, each carrying a payload of arbitrary type and up to 2^{32}-1 octets in size. Records can be chained together to support larger payloads. An NDEF record carries three parameters for describing its payload: the payload length, the payload type, and an optional payload identifier. The purpose of these parameters is as follows:

The payload length

The payload length indicates the number of octets in the payload (see section 2.4.1). By providing the payload length within the first 8 octets of a record, efficient record boundary detection is possible.

The payload type

The NDEF payload type identifier indicates the type of the payload. NDEF supports URIs [RFC 3986], MIME media type constructs [RFC 2046], and an NFC-specific type format as type identifiers (see section 2.4.2). By indicating the type of a payload, it is possible to dispatch the payload to the appropriate user application.

The payload identifier

A payload may be given an optional identifier in the form of an absolute or relative URI (see section 2.4.3). The use of an identifier enables payloads that support URI linking technologies to cross-reference other payloads.

2.2 Intended Usage

The intended usage of NDEF is as follows: A user application wants to encapsulate one or more related documents into a single NDEF message. For example, this can be an application-specific message along with a set of attachments, each of standardized type. The NDEF generator encapsulates each document in NDEF records as payload or chunked payload, indicating the type and length of the payload along with an optional identifier. The NDEF records are then put together to form a single NDEF message. The NDEF message is transmitted across an NFC link to another NFC Forum Device where they are received and parsed, or as an intermediate step, the message is written to an NFC Forum Tag. An NFC Forum Device brought close to this NFC Forum Tag will read the NDEF message from this tag and hand it over to the NDEF parser. The NDEF parser deconstructs the NDEF message and hands the payloads to a (potentially different) user application. Each NDEF message MUST be sent or received in its entirety.

NDEF records can encapsulate documents of any type. It is possible to carry MIME messages in NDEF records by using a media type such as “message/rfc822”. An NDEF message can be encapsulated in an NDEF record by using an NFC-specific predefined type (see [NFC RTD]).

It is important to note that although MIME entities are supported, there are no assumptions in NDEF that a record payload is MIME; NDEF makes no assumption concerning the types of the payloads carried in an NDEF message. Said differently, an NDEF parser need not inspect the NDEF record type nor peer inside an NDEF record in order to parse the NDEF message.

NDEF provides no support for error handling. It is up to the NDEF parser to determine the implications of receiving a malformed NDEF message or an NDEF message containing a field length beyond its processing capabilities. It is the responsibility of the user applications involved to provide any additional functionality such as QoS that they may need as part of the overall system in which they participate.

2.3 NDEF Encapsulation Constructs

2.3.1 Message

An NDEF message is composed of one or more NDEF records. The first record in a message is marked with the *MB* (Message Begin) flag set and the last record in the message is marked with the *ME* (Message End) flag set (see sections 3.2.1 and 3.2.2). The minimum message length is one record which is achieved by setting both the *MB* and the *ME* flag in the same record. Note that at least two record chunks are required in order to encode a chunked payload (see section 2.3.3). The maximum number of NDEF records that can be carried in an NDEF message is unbounded.

NDEF messages MUST NOT overlap; that is, the *MB* and the *ME* flags MUST NOT be used to nest NDEF messages. NDEF messages MAY be nested by carrying a full NDEF message as a payload within an NDEF record.

NDEF Message						
R_1 MB=1	…	R_r	…	R_s	…	R_t ME=1

Figure 1. Example of an NDEF Message with a Set of Records

The message head is to the left and the tail to the right, with the logical record indices t > s > r > 1. The *MB* (Message Begin) flag is set in the first record (index 1) and the *ME* (Message End) flag is set in the last record (index t).

Actual NDEF records do not carry an index number; the ordering is implicitly given by the order in which the records are serialized. For example, if records are repackaged by an intermediate application, then that application is responsible for ensuring that the order of records is preserved.

2.3.2 Record

A record is the unit for carrying a payload within an NDEF message. Each payload is described by its own set of parameters (see section 2.4).

2.3.3 Record Chunks

A record chunk carries a chunk of a payload. Chunked payloads can be used to partition dynamically generated content or very large entities into multiple subsequent record chunks serialized within the same NDEF message.

Chunking is not a mechanism for introducing multiplexing or data streaming into NDEF and it MUST NOT be used for those purposes. It is a mechanism to reduce the need for outbound buffering on the generating side. This is similar to the message chunking mechanism defined in HTTP/1.1 [RFC 2616].

An NDEF message can contain zero or more chunked payloads. Each chunked payload is encoded as an *initial* record chunk followed by zero or more *middle* record chunks and finally by a *terminating* record chunk. Each record chunk is encoded as an NDEF record using the following encoding rules:

- The *initial* record chunk is an NDEF record with the *CF* (Chunk Flag) flag set (see section 3.2.3). The type of the entire chunked payload MUST be indicated in the TYPE field regardless of whether the PAYLOAD_LENGTH field value is zero or not. The ID field MAY be used to carry an identifier of the entire chunked payload. The PAYLOAD_LENGTH field of this initial record indicates the size of the data carried in the PAYLOAD field of the initial record only, not the entire payload size (see section 2.4.1).
- Each *middle* record chunk is an NDEF record with the *CF* flag set indicating that this record chunk contains the next chunk of data of the same type and with the same identifier as the initial record chunk. The value of the TYPE_LENGTH and the *IL* fields MUST be zero and the TNF (Type Name Format) field value MUST be 0x06 *(Unchanged)* (see section 3.2.6). The PAYLOAD_LENGTH field indicates the size of the data carried in the PAYLOAD field of this single middle record only (see section 2.4.1).
- The *terminating* record chunk is an NDEF record with the *CF* flag cleared, indicating that this record chunk contains the last chunk of data of the same type and with the same identifier as the initial record chunk. As with the middle record chunks, the value of the TYPE_LENGTH and the *IL* fields MUST be zero and the TNF (Type Name Format) field value MUST be 0x06 *(Unchanged)* (see section 3.2.6). The PAYLOAD_LENGTH field indicates the size of the data carried in the PAYLOAD field of this terminating record chunk (see section 2.4.1).

A chunked payload MUST be entirely encapsulated within a single NDEF message. That is, a chunked payload MUST NOT span multiple NDEF messages. As a consequence, neither an initial nor a middle record chunk can have the *ME* (Message End) flag set.

2.4 NDEF Payload Description

Each record contains information about the payload carried within it. This section introduces the mechanisms by which these payloads are described.

2.4.1 Payload Length

Regardless of the relationship of a record to other records, the payload length always indicates the length of the payload encapsulated in *this* record. The length of the payload is indicated in the PAYLOAD_LENGTH field. The PAYLOAD_LENGTH field is one octet for *short records* and four octets for normal records. Short records are indicated by setting the *SR* bit flag to a value of 1 (see section 3.2.4). Zero is a valid payload length.

2.4.2 Payload Type

The payload type of a record indicates the kind of data being carried in the payload of that record. This may be used to guide the processing of the payload at the discretion of the user application. The type of the first record, by convention, SHOULD provide the processing context not only for the first record but for the whole NDEF message. Additional context for processing the message MAY be provided, for example, by the link layer service access point (LSAP) or transport service port (e.g. TCP, UDP, etc) at which the message was received and by other communication parameters.

It is important to emphasize that NDEF mandates no specific processing model for NDEF messages. The usage of the payload types is entirely at the discretion of the user application. The comments regarding usage above should be taken as guidelines for building processing conventions, including mappings of higher level application semantics onto NDEF.

The format of the TYPE field value is indicated using the *TNF* (Type Name Format) field (see section 3.2.6). This specification supports TYPE field values in the form of NFC Forum well-known types, NFC Forum external types, absolute URIs [RFC 3986], and MIME media-type constructs. The first allows for NFC Forum specified payload types supporting NFC Forum reference applications [NFC RTD]; URIs provide for decentralized control of the value space; media types allow NDEF to take advantage of the media type value space maintained by IANA [RFC 1700].

The media type registration process is outlined in RFC 2048 [RFC 2048]. Use of non-registered media types is discouraged. The URI scheme registration process is described in RFC 2717 [RFC 2717]. It is RECOMMENDED that only well-known URI schemes registered by IANA be used (see [URI SCHEME] for a current list).

URIs can be used for message types that are defined by URIs. Records that carry a payload with an XML-based message type MAY use the XML namespace identifier of the root element as the TYPE field value. A SOAP/1.1 message, for example, may be represented by the URI

    http://schemas.xmlsoap.org/soap/envelope/

NOTE: Encoding of URI characters which fall outside the US-ASCII range is left to the NDEF application. Therefore, an NDEF parser must not assume any particular encoding for this field. See [RFC 3986] and the specifications of particular protocol schemes (e.g. HTTP, URN, etc.) for more information on parsing of URIs and character encoding requirements for non-ASCII characters.

Records that carry a payload with an existing, registered media type SHOULD carry a TYPE field value of that media type. The TYPE field indicates the type of the payload; it does NOT refer to a MIME message that contains an entity of the given type. For example, the media type

    image/jpeg

indicates that the payload is an image in JPEG format using JFIF encoding as defined by RFC 2046 [RFC 2046]. Similarly, the media type

    message/http

indicates that the payload is an HTTP message as defined by RFC 2616 [RFC 2616]. The value

    application/xml; charset="utf-16"

indicates that the payload is an XML document as defined by RFC 3023 [RFC3023].

2.4.3 Payload Identification

The optional payload identifier allows user applications to identify the payload carried within an NDEF record. By providing a payload identifier, it becomes possible for other payloads supporting URI-based linking technologies to refer to that payload. NDEF does not mandate any particular linking mechanism or format but leaves this to the user application to define in the language it prefers.

It is important that payload identifiers are maintained so that references to those payloads are not broken. If records are repackaged, for example, by an intermediate application, then that application is responsible for ensuring that the linked relationship between identified payloads is preserved.

2.5 NDEF Mechanisms Test Requirements

This section identifies the testable requirements of the NDEF mechanisms defined in chapter 2. The purpose of this section and the table below is to guide the development of conformance tests and does not supersede the normative requirements presented in the other sections of this chapter.

Test Requirements 1. NDEF Mechanisms Test Requirements

Message requirements
Each NDEF message MUST be exchanged in its entirety.
The first record in a message is marked with the *MB* (Message Begin) flag set.
The last record in the message is marked with the *ME* (Message End) flag set.
NDEF messages MUST NOT overlap; that is, the *MB* and the *ME* flags MUST NOT be used to nest NDEF messages.
Record chunk requirements
Each chunked payload is encoded as an initial record chunk followed by 0 or more middle record chunks and finally by a terminating record chunk.
The initial record chunk is an NDEF record with the *CF* (Chunk Flag) flag set.
The type of the entire chunked payload MUST be indicated in the TYPE field of the initial record chunk.
The PAYLOAD_LENGTH field of the initial record indicates the size of the data carried in the PAYLOAD field of the initial record only, not the entire payload size.
Each middle record chunk is an NDEF record with the *CF* flag set.
For each middle record chunk the value of the TYPE_LENGTH and the *IL* fields MUST be 0.
For each middle record chunk the TNF (Type Name Format) field value MUST be 0x06 (*Unchanged*).
For each middle record chunk, the PAYLOAD_LENGTH field indicates the size of the data carried in the PAYLOAD field of this single record only.
The terminating record chunk is an NDEF record with the *CF* flag cleared.
For the terminating record chunk, the value of the TYPE_LENGTH and the *IL* fields MUST be 0.
For the terminating record chunk, the TNF (Type Name Format) field value MUST be 0x06 (*Unchanged*).
For the terminating record chunk, the PAYLOAD_LENGTH field indicates the size of the data carried in the PAYLOAD field of this record only.
A chunked payload MUST be entirely encapsulated within a single NDEF message.
An initial record chunk MUST NOT have the *ME* (Message End) flag set.
A middle record chunk MUST NOT have the *ME* (Message End) flag set.

NDEF payload requirements
The PAYLOAD_LENGTH field is four octets for normal records.
The PAYLOAD_LENGTH field is one octet for records with an *SR* (Short Record) bit flag value of 1.
The PAYLOAD_LENGTH field of a short record MUST have a value between 0 and 255.
The PAYLOAD_LENGTH field of a normal record MUST have a value between 0 and 2^{32}-1.

3 The NDEF Specification

3.1 Data Transmission Order

The order of transmission of the NDEF record described in this document is resolved to the octet level. For diagrams showing a group of octets, the order of transmission of those octets is first left to right and then top to bottom, as they are read in English. For example, in the diagram in Figure 2, the octets are transmitted in the order they are numbered.

```
+--+--+--+--+--+--+--+--+
|        Octet 1        |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|        Octet 2        |        Octet 3        |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|        Octet 4        |
+--+--+--+--+--+--+--+--+
|        Octet 5        |
+--+--+--+--+--+--+--+--+
```

Figure 2. NDEF Octet Ordering

Whenever an octet represents a numeric quantity, the leftmost bit in the diagram is the high order or most significant bit. For each multi-octet field representing a numeric quantity defined by NDEF, the leftmost bit of the whole field is the most significant bit. Such quantities are transmitted in a big-endian manner with the most significant octet transmitted first.

3.2 Record Layout

NDEF records are variable length records with a common format illustrated in the figure below. In the following sections, the individual record fields are described in more detail.

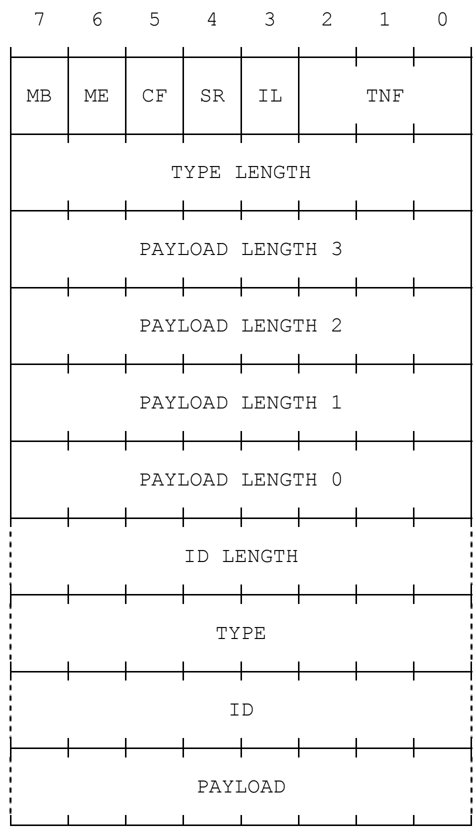

Figure 3. NDEF Record Layout

3.2.1 MB (Message Begin)

The *MB* flag is a 1-bit field that when set indicates the start of an NDEF message (see section 2.3.1).

3.2.2 ME (Message End)

The *ME* flag is a 1-bit field that when set indicates the end of an NDEF message (see section 2.3.1). Note, that in case of a chunked payload, the *ME* flag is set only in the terminating record chunk of that chunked payload (see section 2.3.3).

3.2.3 CF (Chunk Flag)

The *CF* flag is a 1-bit field indicating that this is either the first record chunk or a middle record chunk of a chunked payload (see section 2.3.3 for a description of how to encode a chunked payload).

3.2.4 SR (Short Record)

The *SR* flag is a 1-bit field indicating, if set, that the PAYLOAD_LENGTH field is a single octet. This *short record* layout is intended for compact encapsulation of small payloads which will fit within PAYLOAD fields of size ranging between 0 to 255 octets.

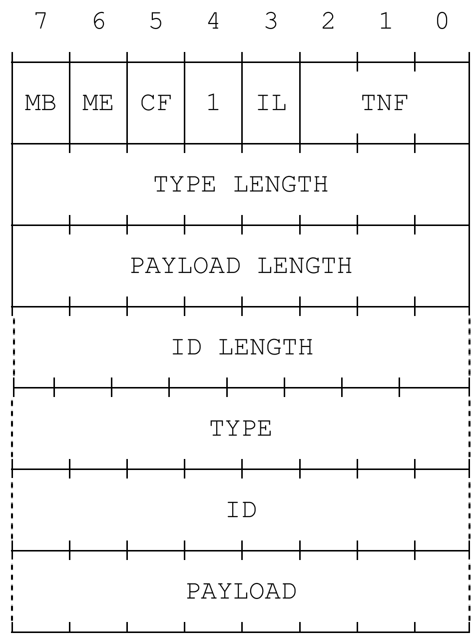

Figure 4. NDEF Short-Record Layout (SR=1)

While it is tempting for implementers to choose one or the other record layout exclusively for a given application, NDEF parsers MUST accept both normal and *short record* layouts. NDEF generators MAY generate either record layout as they deem appropriate. A single NDEF message MAY contain both normal and short records.

3.2.5 IL (ID_LENGTH field is present)

The *IL* flag is a 1-bit field indicating, if set, that the ID_LENGTH field is present in the header as a single octet. If the *IL* flag is zero, the ID_LENGTH field is omitted from the record header and the ID field is also omitted from the record.

3.2.6 TNF (Type Name Format)

The *TNF* field value indicates the structure of the value of the TYPE field (see section 2.4.2 for a description of the TYPE field and section 4 for a description of internationalization issues related to the TYPE field). The TNF field is a 3-bit field with values defined in the table below:

Table 1. TNF Field Values

Type Name Format	Value
Empty	0x00
NFC Forum well-known type [NFC RTD]	0x01
Media-type as defined in RFC 2046 [RFC 2046]	0x02
Absolute URI as defined in RFC 3986 [RFC 3986]	0x03
NFC Forum external type [NFC RTD]	0x04
Unknown	0x05
Unchanged (see section 2.3.3)	0x06
Reserved	0x07

The value 0x00 (*Empty*) indicates that there is no type or payload associated with this record. When used, the TYPE_LENGTH, ID_LENGTH, and PAYLOAD_LENGTH fields MUST be zero and the TYPE, ID, and PAYLOAD fields are thus omitted from the record. This TNF value can be used whenever an empty record is needed; for example, to terminate an NDEF message in cases where there is no payload defined by the user application.

The value 0x01 (*NFC Forum well-known type*) indicates that the TYPE field contains a value that follows the *RTD* type name format defined in the NFC Forum RTD specification [NFC RTD].

The value 0x02 (*media-type*) indicates that the TYPE field contains a value that follows the *media-type* BNF construct defined by RFC 2046 [RFC 2046].

The value 0x03 (*absolute-URI*) indicates that the TYPE field contains a value that follows the *absolute-URI* BNF construct defined by RFC 3986 [RFC 3986].

The value 0x04 (*NFC Forum external type*) indicates that the TYPE field contains a value that follows the type name format defined in [NFC RTD] for external type names.

The value 0x05 (*Unknown*) SHOULD be used to indicate that the type of the payload is unknown. This is similar to the "application/octet-stream" media type defined by MIME [RFC 2046]. When used, the TYPE_LENGTH field MUST be zero and thus the TYPE field is omitted from the NDEF record. Regarding implementation, it is RECOMMENDED that an NDEF parser receiving an NDEF record of this type, without further context to its use, provides a mechanism for storing but not processing the payload (see section 4.2).

The value 0x06 (*Unchanged*) MUST be used in all middle record chunks and the terminating record chunk used in chunked payloads (see section 2.3.3). It MUST NOT be used in any other record. When used, the TYPE_LENGTH field MUST be zero and thus the TYPE field is omitted from the NDEF record.

There is no default value for the TNF field. Reserved (or unassigned) TNF field values are for future use and MUST NOT be used. An NDEF parser that receives an NDEF record with an unknown or unsupported TNF field value SHOULD treat it as 0x05 (*Unknown*).

3.2.7 TYPE_LENGTH

The *TYPE_LENGTH* field is an unsigned 8-bit integer that specifies the length in octets of the TYPE field. The TYPE_LENGTH field is always zero for certain values of the TNF field (see section 3.2.6).

3.2.8 ID_LENGTH

The *ID_LENGTH* field is an unsigned 8-bit integer that specifies the length in octets of the ID field. This field is present only if the *IL* flag is set to 1 in the record header. An ID_LENGTH of zero octets is allowed and, in such cases, the ID field is omitted from the NDEF record.

3.2.9 PAYLOAD_LENGTH

The *PAYLOAD_LENGTH* field is an unsigned integer that specifies the length in octets of the PAYLOAD field (the application payload). The size of the PAYLOAD_LENGTH field is determined by the value of the *SR* flag (see section 3.2.4).

If the *SR* flag is set, the PAYLOAD_LENGTH field is a single octet representing an 8-bit unsigned integer.

If the *SR* flag is clear, the PAYLOAD_LENGTH field is four octets representing a 32-bit unsigned integer. Transmission order of the octets is MSB-first (see section 3.1).

A payload length of 0 is allowed in which case the PAYLOAD field is omitted from the NDEF record. Application payloads larger than 2^{32}-1 octets can be accommodated by using chunked payloads (see section 2.3.3).

3.2.10 TYPE

The value of the *TYPE* field is an identifier describing the type of the payload (see section 2.4.2). The value of the TYPE field MUST follow the structure, encoding, and format implied by the value of the TNF field (see section 3.2.6).

An NDEF parser receiving an NDEF record with a TNF field value that it supports but an unknown TYPE field value SHOULD interpret the type identifier of that record as if the TNF field value were 0x05 (*Unknown*).

It is STRONGLY RECOMMENDED that the type identifier be globally unique and maintained with stable and well-defined semantics over time.

3.2.11 ID

The value of the ID field is an identifier in the form of a URI reference [RFC 3986] (see sections 2.4.3 and 4.4). The required uniqueness of the message identifier is guaranteed by the generator. The URI reference can be either relative or absolute; NDEF does not define a base URI which means that user applications using relative URIs MUST provide an actual or a virtual base URI (see [RFC 3986]).

Middle and *terminating* record chunks (that is, records containing other than the *initial* chunk of a chunked payload; see section 2.3.3) MUST NOT have an ID field. All other records MAY have an ID field.

3.2.12 PAYLOAD

The PAYLOAD field carries the payload intended for the NDEF user application. Any internal structure of the data carried within the PAYLOAD field is opaque to NDEF.

3.3 THE NDEF Specification Test Requirements

This section identifies the testable requirements of the NDEF mechanisms defined in chapter 3. The purpose of this section is to guide the development of conformance tests and does not supersede the normative requirements presented in the other sections of this chapter.

Test Requirements 2. The NDEF Specification Test Requirements

Data transmission order requirements
Quantities are transmitted in a big-endian manner with the most significant octet transmitted first.
Record layout requirements
NDEF parsers MUST accept both normal and short record layouts.
NDEF parsers MUST accept single NDEF messages composed of both normal and short records.
If the IL flag is 1, the ID_LENGTH field MUST be present.
If the IL flag is 0, the ID_LENGTH field MUST NOT be present.
If the IL flag is 0, the ID field MUST NOT be present.
The TNF field MUST have a value between 0x00 and 0x06.
If the TNF value is 0x00, the TYPE_LENGTH, ID_LENGTH, and PAYLOAD_LENGTH fields MUST be zero and the TYPE, ID, and PAYLOAD fields MUST be omitted from the record.
If the TNF value is 0x05 (Unknown), the TYPE_LENGTH field MUST be 0 and the TYPE field MUST be omitted from the NDEF record.
If the TNF value is 0x06 (Unchanged), the TYPE_LENGTH field MUST be 0 and the TYPE field MUST be omitted from the NDEF record.
The TNF value MUST NOT be 0x07.
If the ID_LENGTH field has a value 0, the ID field MUST NOT be present.
If the SR flag is 0, the PAYLOAD_LENGTH field is four octets, representing a 32-bit unsigned integer, and the transmission order of the octets is MSB-first.
If the SR flag is 1, the PAYLOAD_LENGTH field is a single octet representing an 8-bit unsigned integer.
If the PAYLOAD_LENGTH field value is 0, the PAYLOAD field MUST NOT be present.
The value of the TYPE field MUST follow the structure, encoding, and format implied by the value of the TNF field.
Middle and terminating record chunks MUST NOT have an ID field.

4 Special Considerations

4.1 Internationalization

Identifiers used in NDEF such as URIs and MIME media-type constructs may provide different levels of support for internationalization. Implementers are referred to RFC 2718 [RFC 2718] for internationalization considerations of URIs, RFC 2046 [RFC 2046] for internationalization considerations of MIME media types and RFC 2047 [RFC 2047] for internationalization of message headers (MIME).

4.2 Security

Implementers should pay special attention to the security implications of any record types that can cause the remote execution of any actions in the recipient's environment. Before accepting records of any type, an application should be aware of the particular security implications associated with that type.

Security considerations for media types in general are discussed in RFC 2048 [RFC 2048] and in the context of the "application" media types in RFC 2046 [RFC 2046].

4.3 Maximum Field Sizes

The size of the PAYLOAD field and the values used in the ID field and the TYPE field are limited by the maximum size of these fields. The maximum size of the PAYLOAD field is 2^{32}-1 octets in the normal NDEF record layout and 255 octets in the *short record* layout (see section 3.2.4). The maximum size of values in the ID and TYPE fields is 255 octets in both record layouts.

While messages formed to these maximal record and field sizes are considered well-formed, not all user applications will have the ability or the need to handle payload content, payload IDs, or types identifiers of these maximal sizes. NDEF parsers that are resource-constrained MAY choose to reject messages that are not sized to fit their specific needs.

However, NDEF parsers MUST NOT reject an NDEF message based solely on the value of the *SR* flag.

4.4 Use of URIs in NDEF

NDEF uses URIs [RFC 3986] for some identifiers. To NDEF, a URI is simply a formatted string that identifies—via name, location, or any other characteristic—a resource.

The use of IP addresses in URIs SHOULD be avoided whenever possible (see RFC 1900 [RFC 1900]). However, when used, the literal format for IPv6 addresses in URIs as described by RFC 2732 [RFC 2732] SHOULD be supported.

NDEF does not define any equivalence rules for URIs in general as these are defined by the individual URI schemes and by RFC 3986 [RFC 3986]. However, because of inconsistencies with respect to some URI equivalence rules in many current URI parsers, it is RECOMMENDED that generators of NDEF messages rely only on the most rudimentary equivalence rules defined by RFC 3986.

4.5 Special Consideration Test Requirements

This section identifies the testable requirements of the NDEF mechanisms defined in chapter 4. The purpose of this section and the table below is to guide the development of conformance tests and does not supersede the normative requirements presented in the other sections of this chapter.

Test Requirements 3. Special Consideration Test Requirements

An NDEF parser MUST NOT reject an NDEF message based solely on the value of the SR flag.
An NDEF parser MAY reject messages that include records with TYPE, ID, or PAYLOAD fields larger than its design limits.

A. Revision History

The following table outlines the revision history of the NDEF Technical Specification.

Table 2. Revision History

Document Name	Revision and Release Date	Status	Change notice	Supersedes
NFCForum-TS-NDEF_1.0	1.0, July 2006	Final	none	

NFC Record Type Definition (RTD)

Technical Specification

NFC Forum[TM]

RTD 1.0

NFCForum-TS-RTD_1.0

2006-07-24

NFC Forum, Inc.
401 Edgewater Place, Suite 600
Wakefield, MA, USA 01880

Contents

Figures

Tables

Test Requirements

1 Introduction

The International Standard ISO/IEC 18092, Near Field Communication - Interface and Protocol (NFCIP-1), defines an interface and protocol for simple wireless interconnection of closely coupled devices operating at 13.56 MHz.

The NFC Data Exchange Format (NDEF) specification defines a data format to exchange information between an NFC Forum Device and another NFC Forum Device or an NFC Forum Tag. The information that can be exchanged by means of NDEF may describe which services an NFC Forum Device or NFC Forum Tag offers, it may contain application or service-specific parameters and meta-data, or it may describe capabilities of NFC Forum Devices or NFC Forum Tags.

NDEF supports the use of standardized MIME content types and URIs to describe record content which is specified outside of the NFC Forum. This specification describes two NFC Forum specific types, known as “NFC Forum Well Known Types” and “NFC external types”.

1.1 Objectives

The NFC Data Exchange Format (NDEF) specification is a common data format for NFC Forum Devices.

The NFC Data Exchange Format specification defines the NDEF data structure format as well as rules to construct a valid NDEF packet as a collection of NDEF records. Furthermore, it defines the mechanism for constructing unique NDEF record type names by different parties, including a format for NFC Forum well-known types.

The NDEF specification defines only the data structure format to exchange application or service specific data in an interoperable way, and it does not define any record types in detail. Specific record types are defined in separate documents.

The first part of this specification considers the type format of the NFC Forum well-known types—that is, the contents of an NDEF Type field when the “TNF” header field value is 0x01 (NFC Well-known Type).

The second part of this specification considers the third party extension type known as an “NFC external type”, which is signified by the TNF header field value of 0x04.

1.2 Purpose

1.2.1 Mission Statement and Goals

It is the mission of the NFC Forum to ensure interoperability of the NFC technology in a broad variety of devices. The RTD specification is intended to support NFC-specific application and service frameworks by providing a means for reservation of well-known record types, and third party extension types.

The RTD specification provides guidelines for the specification of well-known types for inclusion in NDEF messages exchanged between NFC Forum devices and between NFC Forum Devices and NFC Forum Tags.

Actual type registrations are not provided in this specification but are expected to be included in other documents.

1.3 References

[ASCII]	ANSI X3.4-1986, Coded Character Set 7-bit American Standard Code for Information Interchange
[ISO/IEC 18092]	Information Technology- Telecommunications and information exchange between systems- Near Field Communication - Interface and Protocol (NFCIP-1).
[NDEF]	NFC Data Exchange Format, NFC Forum, 2006.
[NFC Best]	Best Practices for NFC Forum Terminology, NFC Forum, Technical Committee Document.
[RFC 2119]	S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Harvard University, March 1997. http://www.apps.ietf.org/rfc/rfc2119.html
[RFC 2046]	N. Freed, N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types" RFC 2046, Innosoft, First Virtual, November 1996.
[RFC 2141]	R. Moats, "URN SYNTAX", May 1997.
[RFC 2234]	D. Crocker, P. Overell, "Augmented BNF for Syntax Specifications: ABNF", November 1997.
[RFC 3986]	T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 3986, MIT/LCS, U.C. Irvine, Xerox Corporation, January 2005. http://www.apps.ietf.org/rfc/rfc3986.html

1.4 Administration

The NFC Record Type Definition (RTD) Specification is an open specification supported by the Near Field Communication Forum, Inc., located at:

401 Edgewater Place, Suite 600
Wakefield, MA, 01880

Tel.: +1 781-876-8955
Fax: +1 781-224-1239

http://www.nfc-forum.org

The Reference Applications Framework technical working group maintains this specification.

This specification has been contributed to by Microsoft, Nokia, Panasonic, Philips, and Sony.

1.5 Special Word Usage

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

1.6 Name and Logo Usage

The Near Field Communication Forum's policy regarding the use of the trademarks *NFC Forum* and the NFC Forum logo is as follows:

- Any company MAY claim compatibility with NFC Forum specifications, whether a member of the NFC Forum or not.
- Permission to use the NFC Forum logos is automatically granted to designated members only as stipulated on the most recent Membership Privileges document, during the period of time for which their membership dues are paid.
- Member's distributors and sales representatives MAY use the NFC Forum logo in promoting member's products sold under the name of the member.
- The logo SHALL be printed in black or in color as illustrated on the Logo Page that is available from the NFC Forum at the address above. The aspect ratio of the logo SHALL be maintained, but the size MAY be varied. Nothing MAY be added to or deleted from the logos.
- Since the NFC Forum name is a trademark of the Near Field Communication Forum, the following statement SHALL be included in all published literature and advertising material in which the name or logo appears:

  NFC Forum and the NFC Forum logo are trademarks of the Near Field Communication Forum.

1.7 Intellectual Property

The NFC Record Type Definition (RTD) Specification conforms to the Intellectual Property guidelines specified in the NFC Forum's Intellectual Property Right Policy, as approved on November 9, 2004 and outlined in the NFC Forum Rules of Procedures, as approved on December 17, 2004.

1.8 Glossary

This section defines all relevant terms and acronyms used in this specification.

Table 1. Definitions

NDEF application

> The logical, higher-layer application on an NFC Forum Device using NDEF to format information for exchange with other NFC Forum Devices or NFC Forum Tags. Also *user application* or *NDEF user application*.

NDEF message

> The basic message construct defined by this specification. An NDEF message contains one or more NDEF records.

NDEF record

> An NDEF record contains a payload described by a type, a length, and an optional identifier.

NDEF payload

The application data carried within an NDEF record.

NDEF generator

An entity or module that encapsulates application-defined payloads within NDEF messages.

NDEF parser

An entity or module that parses NDEF messages and hands off the payloads to an NDEF application.

User Application

See *NDEF Application*.

Table 2. Acronyms

NDEF	NFC Data Exchange Format. See *NFC DATA EXCHANGE FORMAT (NDEF*, Re-Draft Revision 0.96*)*, NFC Forum Draft, October 2005
NID	Namespace Identifier. Identifies uniquely an URN namespace. Please see [RFC 2141] for a full definition.
NSS	Namespace Specific String. The rest of the URN after the NID. See [RFC 2141] for a full definition.
MIME	Multipurpose Internet Mail Extensions. A standard specifying the format of strongly-typed data transferred over the Internet. Defined in [RFC 2045-2049]
RTD	Record Type Definition. An NFC-specific record type and type name which may be carried in an NDEF record with a TNF field value of 0x01 (NFC Well-Known Type).
URI	Uniform Resource Identifier. A compact sequence of characters that identifies an abstract or physical resource. [RFC 3986] Uniform Resource Names (URNs) and Uniform Resource Locators (URLs) are both forms of URI.
URN	Uniform Resource Name. A particular type of URI that is defined in [RFC 2141].

2 Record Types

The record type string field of an NDEF record contains the name of the record type (called "record type name"). Record type names are used by NDEF applications to identify the semantics and structure of the record content.

Record type names may be specified in several formats, called Type Name Formats, as signified by the TNF field of the NDEF record header. Record type names may be MIME media types, absolute URIs, NFC Forum external type names, or may be well-known NFC type names (RTD's, the subject of this specification).

Each record type definition is identified by its record type name.

Record type names can be defined by the NFC Forum and by third parties. In the following sections, the rules governing the RTD type name space are defined.

2.1 NFC Forum Well-known Type

The NFC Forum Well-known Type is a dense format designed for tags and creating primitives for certain common types. It is meant to be used in case there is no equivalent URI or MIME type available, or when message size limitations require a very short name.

An NFC Forum Well Known Type is identified inside an NDEF message by setting the TNF field of a record to the value of 0x01, as defined in the NDEF specification.

An NFC Forum Well-Known Type is a URN as defined by [RFC 2141], with the namespace identifier (NID) "nfc".

The Namespace Specific String of the NFC Well Known Type URN is prefixed with "wkt:". However, when encoded in an NDEF message, the Well Known Type MUST be written as a relative-URI construct [RFC 3896], omitting the NID and the "wkt:" –prefix.

For example, the Well Known Type "urn:nfc:wkt:a" would be encoded as "a". The Well Known Type "urn:nfc:wkt:Very-complicated-type" would be encoded as "Very-complicated-type".

There are two kinds of NFC Forum Well Known Types detailed in the sections below. For brevity, we exclude the URN NID and the NSS prefix from the examples.

For a definition of the character ranges used in the Well Known Types, please see chapter 3.

2.1.1 NFC Forum Global Type

The NFC Forum is responsible for defining and managing NFC Forum Global Types. Other parties MUST NOT define or redefine these.

An NFC Forum Global Type SHALL start with an upper-case letter (character range <upper>).

Examples of NFC Forum Global Types: "U", "Cfq", "Trip-to-Texas".

2.1.2 NFC Forum Local Type

NFC Forum Local Types SHALL start with a character in sets <lower> or <number>.

NFC Forum Local Types are available for use within the context of another record. A processing application MUST NOT process these types when application context is not available. Local types are used whenever the burden of using a long, domain name–based external type is too much, and there is no need to define its meaning outside of the local context.

An RTD or an application defines the context for the interpretation for a Local Type. A Local Type MAY be reused by another application in a different context and with different content.

Examples of NFC Forum Local Types: “0”, “foo”, “u”.

2.2 NFC Forum External Type

The External Type Name is meant for organizations that wish to self-allocate a name space to be used for their own purposes.

An External Type is identified in an NDEF record by setting the TNF field value to 0x04, as defined in the NDEF specification [NDEF].

The External Type is, much like a Well Known Type, an URN, with the NID of “nfc”. However, the NSS specific part is put into another namespace named “ext”. A canonical version of the External Type Name would look like:

“urn:nfc:ext:example.com:f”

The External Type Name MUST be formed by taking the domain name of the issuing organization, adding a colon, and then adding the type name as managed by the organization.

As with Well Known Types, the binary encoding of External Type Name inside NDEF messages MUST omit the NID and the NSS prefix of “ext”.

2.3 Record Types Generic Requirements

Test Requirements 1. Record Types Generic Requirements

NFC Forum standardized types defined as RTD records SHALL use NFC Forum Well-Known type names.
When packaged into NDEF records, NFC Forum standardized types defined as RTD records SHALL be signified in the NDEF record header by the Type Name Format (TNF) field value of 0x01 (NFC Forum Well-Known Type).
An NFC Forum Well Known Type SHALL be a URN with the “urn:nfc:wkt:” prefix.
An NFC Forum Global Type MUST NOT be defined or redefined by other parties than NFC Forum.
An NFC Forum Global Type SHALL start with a character in the range <upper> as defined in Chapter 3.
An NFC Forum Local Type SHALL start with a character in the range <lower> or <number> as defined in Chapter 3.
A processing application MUST NOT process a NFC Forum Local Type if an application context is not available.
An NFC Forum Local Type MAY be reused by another application in a different context and with different content.
An NFC Forum External Type SHALL be identified with the TNF field value of 0x04.
An NFC Forum External Type SHALL be a URN with the prefix of “urn:nfc:ext:”.
In the NDEF binary format, the URN prefix MUST NOT be used.
The External Type MUST be formed by taking the domain name of the issuing organization, adding a colon, and then adding a type name. An External Type MUST include a colon and a non-zero length type name.

3 RTD Type Names

This section defines the normative requirements for the NFC Forum Well-Known Type Names (below: RTD-URI). The language used is the ABNF format as defined in RFC 2234 [RFC 2234].

```
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          = “%” / “/” / “?” / “#”
```

3.1 Binary Encoding

The binary encoding of Well Known Types and External Type Names for NDEF MUST be done according to the ASCII chart in Appendix A.

The URN NID and the NFC NSS prefixes MUST NOT be included in the binary NDEF format. (However, if RTDs are used in other formats, such as XML, the URNs SHOULD be given in the absolute URN format.)

NOTE: *This specification does not define legal characters for any particular record content. Record content is specified in other documents, specific to those record types.*

3.2 Percent Encoding in NFC Forum Types

To help define equivalence rules for NFC Forum Well Known Types, NFC Forum will not issue a Global Type Name using percent-encoding as defined in [RFC 2141]. Any Local Type Name used by third parties MUST NOT use the percent encoding.

External Types SHOULD NOT use the percent encoding. However, an application using such an external type MUST first encode the string in UTF-8 before converting it to the percent encoding.

3.3 Equivalence of Record Type Names

The comparison of record type names is done on a character-by-character basis.

Two Well Known Type names MUST be compared in a case-sensitive manner. Because of the fact that the encoding is fixed to US-ASCII, it also implies that two Well Known Types MUST be considered equivalent if and only if their binary representations are identical.

Example:

```
"Foobar"
"fooBar"
"fOoBaR"
"foobar"
```

The four examples above are all different Well Known Type names.

Two External Type Names MUST be compared in a case-insensitive manner. Example:

```
"example.com:foobar"
"Example.com:foobar"
"Example.COM:Foobar"
"eXaMpLe.CoM:fOoBaR"
```

The four examples above represent all the same External Type Name.

3.4 RTD Type Names Requirements

Test Requirements 2. RTD Type Names Requirements

The binary encoding of Well Known Types (including Global and Local Names) and External Type names MUST be done according to the ASCII chart in Appendix A.
Well Known Types (including Global and Local Names) MUST NOT use the percent-encoding as defined by RFC 2141.
External types SHOULD NOT use the percent encoding as defined by RFC 2141.
Two Well Known Types (including Global and Local Names) MUST be compared on a case-sensitive, character-by-character basis. In other words, two Well Known Types MUST be considered equal if and only if their binary representations are identical.
Two External Types MUST be compared on a case insensitive, character-by-character basis.

4 Error Handling

4.1 Illegal characters

A record with a type name containing characters outside of the valid range of characters defined in Chapter 3 MUST be ignored.

4.2 Unknown Record Types

Applications MUST ignore records which have a Well Known Type or an External Type that they do not recognize.

4.3 Error Handling Requirements

Test Requirements 3. Error Handling Requirements

Any character not defined as a valid character in Chapter 3 SHALL be considered an illegal character in a record type name.
Records containing illegal characters in the record type name MUST be ignored.
An application that does not recognize a record type name MUST ignore the entire record.

A. Character Set for Record Types

Record type names SHALL be formed of characters from of the US ASCII [ASCII] character set. Characters in the range [0-31] and 127 decimal, as shown in the following table, SHALL NOT be used in record type names.

Table 3. ASCII Character Chart

Binary	Dec	Hex	Graph.	Binary	Dec	Hex	Graph.	Binary	Dec	Hex	Graph.
0010 0000	32	20	(blank)	0100 0000	64	40	@	0110 0000	96	60	`
0010 0001	33	21	!	0100 0001	65	41	A	0110 0001	97	61	a
0010 0010	34	22	"	0100 0010	66	42	B	0110 0010	98	62	b
0010 0011	35	23	#	0100 0011	67	43	C	0110 0011	99	63	c
0010 0100	36	24	$	0100 0100	68	44	D	0110 0100	100	64	d
0010 0101	37	25	%	0100 0101	69	45	E	0110 0101	101	65	e
0010 0110	38	26	&	0100 0110	70	46	F	0110 0110	102	66	f
0010 0111	39	27	'	0100 0111	71	47	G	0110 0111	103	67	g
0010 1000	40	28	(	0100 1000	72	48	H	0110 1000	104	68	h
0010 1001	41	29	)	0100 1001	73	49	I	0110 1001	105	69	i
0010 1010	42	2A	*	0100 1010	74	4A	J	0110 1010	106	6A	j
0010 1011	43	2B	+	0100 1011	75	4B	K	0110 1011	107	6B	k
0010 1100	44	2C	,	0100 1100	76	4C	L	0110 1100	108	6C	l
0010 1101	45	2D	-	0100 1101	77	4D	M	0110 1101	109	6D	m
0010 1110	46	2E	.	0100 1110	78	4E	N	0110 1110	110	6E	n
0010 1111	47	2F	/	0100 1111	79	4F	O	0110 1111	111	6F	o
0011 0000	48	30	0	0101 0000	80	50	P	0111 0000	112	70	p
0011 0001	49	31	1	0101 0001	81	51	Q	0111 0001	113	71	q
0011 0010	50	32	2	0101 0010	82	52	R	0111 0010	114	72	r
0011 0011	51	33	3	0101 0011	83	53	S	0111 0011	115	73	s
0011 0100	52	34	4	0101 0100	84	54	T	0111 0100	116	74	t
0011 0101	53	35	5	0101 0101	85	55	U	0111 0101	117	75	u
0011 0110	54	36	6	0101 0110	86	56	V	0111 0110	118	76	v
0011 0111	55	37	7	0101 0111	87	57	W	0111 0111	119	77	w
0011 1000	56	38	8	0101 1000	88	58	X	0111 1000	120	78	x
0011 1001	57	39	9	0101 1001	89	59	Y	0111 1001	121	79	y
0011 1010	58	3A	:	0101 1010	90	5A	Z	0111 1010	122	7A	z
0011 1011	59	3B	;	0101 1011	91	5B	[	0111 1011	123	7B	{
0011 1100	60	3C	<	0101 1100	92	5C	\	0111 1100	124	7C	|
0011 1101	61	3D	=	0101 1101	93	5D	]	0111 1101	125	7D	}
0011 1110	62	3E	>	0101 1110	94	5E	^	0111 1110	126	7E	~
0011 1111	63	3F	?	0101 1111	95	5F	_				

B. Record Type Name Examples

The contents of this appendix are informative and describe examples for encoding and comparing record type names into their binary representation.

An example of translating a record type name into binary representation:

Table 4. Translating Record Type Names into Binary Representation

String Representation	Binary Representation (as hexadecimal)
Sms	53 6D 73
sms	73 6D 73

To encode the binary representation of the type names, each character from the string representation is replaced by its binary value from Appendix A.

In this example, the two record type names are considered non-equivalent since their binary representations are not identical. The case-sense of letters in the string, white space, and other language comparison rules are not considered when comparing type strings for equivalence. Only the binary representations are considered.

C. Discussion on Associating Records

The contents of this appendix are informative.

There are two basic ways to associate NDEF records to each other. The first one is called "association by reference", which is amounts to a flat hierarchy or a list of objects.

When associating records by reference, the context is typically given by the first record in the message. This is the same association model that is used by MIME. For example, if you wish to represent an email message with two PNG attachments as an NDEF message, you first send the email message in one record (typed message/rfc822), then the first PNG image as a separate record (image/png), and the second PNG image (again, image/png). To illustrate:

NDEF MESSAGE		
Email (message/rfc822)	Pic1.png (image/png)	Pic2.png (image/png)

Figure 1. NDEF Messages (Multiple)

This method allows an application to lift the PNG images off the message, even if it does not understand the email message. In general, when designing your own record types, you should choose association by reference if your message parts would be valuable even on their own, i.e., even if the context is not understood. Association by reference is also a good model if you are moving a large amount of data because it allows you to take advantage of the chunking feature of NDEF. In addition, it also allows the processing to start at the receiver end before the message is finished (this is one of the reasons why it is good to declare the context at the beginning of the message).

The second way is called "association by containment". This is a hierarchical model (not entirely unlike XML or HTML), where the content portion of an NDEF records contains an NDEF message. This is very useful in the case where you wish to imply a stronger relationship between records, or need to serialize information that is already in a hierarchical format. Also, if you are going to send multiple objects of the same type within the message, you probably wish to use an containment model, and then string them together in a list(so yes, it is possible and very sensible to mix these models).

For example, the Smart Poster record defines a URI plus some added metadata about that URI. The added metadata is not useful to an application without the URI itself, and in fact, it would be relatively meaningless. To illustrate:

NDEF Message				
Sp (Smart Poster)				application/vcard
URI	Text	Action	Configuration	vCard data

Figure 2. NDEF Message with Metadata

In this case, there are two records in the NDEF message. The first one is a Smart Poster containing a URI, a Text record for a title, an action, and a configuration record; whereas the other one is just a normal vCard (using the vCard standard). (In this case, there is no particular context defined for the vCard, so an application may either ignore it or use it for some purpose; this is an implementation detail. In general, putting records describing different things and assuming some particular context or processing model will probably result in interoperability trouble.)

Anyway, since the Text, Action, and Configuration are so tightly coupled with the URI (the URI might not even be fetchable without the proper configuration, if the config defines a local access point), they work better using a containment model than a reference model.

Neither of these examples displayed any use of the ID field, which can be used in both models with equal efficiency. In association by reference, the first record typically lists the IDs that it uses and defines the context that way; in association by containment, the IDs would typically be used to signify the role of a record (e.g., “A record with an ID of 'config' shall be used for defining an access point.”)

Of course, an application is free to mix-and-match these association types. There is no hard-and-fast rule to say which one is better in a given situation, and as designed, this allows maximum flexibility to the application developer.

A third, but deprecated, practice would be using ordering (i.e., record #1 would always signify something, record #2 something else, record #3 again something else), but this, in general, is not a good idea, since you cannot rely on any particular behavior of a NDEF processor. It could be that by the time your application receives the NDEF message, records may have been inserted or removed. Do not rely on any implementation-specific behavior. This seems obvious to any seasoned developer, but it is easy to forget in the rush of a deadline.

The advice in this discussion is offered because it is likely that developers at some point face the need to associate NDEF records with each other, and it is good that some of the best practices and conventions are laid out for all to see. Reading a new specification can be difficult, and hopefully discussion such as this will ease the work of the developer.

D. Revision History

The following table outlines the revision history of the RTD Technical Specification.

Table 5. Revision History

Document Name	Revision and Release Date	Status	Change notice	Supersedes
NFCForum-TS-RTD_1.0	1.0, July 2006	Final	none	

Text Record Type Definition

Technical Specification

NFC Forum™

RTD-Text 1.0

NFCForum-TS-RTD_Text_1.0

2006-07-24

NFC Forum, Inc.
401 Edgewater Place, Suite 600
Wakefield, MA, USA 01880

Contents

Tables

1 Overview

The Text Record Type Description defines an NFC Forum Well Known Type [NFC RTD] for plain text data. It may be used as free form text descriptions of other objects on an RFID tag.

1.1 Objectives

The objective of this document is to function as a normative reference to the Text RTD.

1.2 Purpose

1.2.1 Mission Statement and Goals

The Text RTD was designed to be used as a general purpose text field to add metadata to things such as URLs. It needs to provide a lightweight component with clearly defined semantics.

The goal is not to replace text/plain, but to define a clear subset that can be used in cases where there is not much space to be used, and to cover the most probable use cases.

The Text RTD must work well for non-western languages also, and it needs to include the language information for localization purposes so that the language can be identified and served to the user.

1.3 References

[NDEF]	“NFC Data Exchange Format Specification”, NFC Forum, 2006.
[NFC RTD]	“NFC Record Type Definition (RTD) Specification”, NFC Forum, 2006.
[RFC 2119]	S. Bradner, “Key words for use in RFCs to Indicate Requirement Levels”, RFC 2119, Harvard University, March 1997. http://www.apps.ietf.org/rfc/rfc2119.html
[RFC 3066]	H. Alvestrand, “Tags for the Identification of Languages”, RFC 3066, Cisco Systems, January 2001. http://www.faqs.org/rfcs/rfc3066.html
[RFC 3066bis]	A. Phillips, M. Davis, “Tags for the Identification of Languages”. IETF Draft. http://www.ietf.org/internet-drafts/draft-ietf-ltru-registry-14.txt
[UNICODE]	“The Unicode 4.0.1 standard”. http://www.unicode.org/versions/Unicode4.0.1/

1.4 Administration

The Text RTD Specification is an open specification supported by the Near Field Communication Forum, Inc., located at:

401 Edgewater Place, Suite 600
Wakefield, MA, 01880

Tel.: +1 781-876-8955
Fax: +1 781-224-1239

http://www.nfc-forum.org

The Reference Applications Framework technical working group maintains this specification.

1.5 Special Word Usage

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119.

1.6 Name and Logo Usage

The Near Field Communication Forum’s policy regarding the use of the trademarks *NFC Forum* and the NFC Forum logo is as follows:

- Any company MAY claim compatibility with NFC Forum specifications, whether a member of the NFC Forum or not.
- Permission to use the NFC Forum logos is automatically granted to designated members only as stipulated on the most recent Membership Privileges document, during the period of time for which their membership dues are paid.
- Member’s distributors and sales representatives MAY use the NFC Forum logo in promoting member’s products sold under the name of the member.
- The logo SHALL be printed in black or in color as illustrated on the Logo Page that is available from the NFC Forum at the address above. The aspect ratio of the logo SHALL be maintained, but the size MAY be varied. Nothing MAY be added to or deleted from the logos.
- Since the NFC Forum name is a trademark of the Near Field Communication Forum, the following statement SHALL be included in all published literature and advertising material in which the name or logo appears:

  NFC Forum and the NFC Forum logo are trademarks of the Near Field Communication Forum.

1.7 Intellectual Property

The Text RTD Specification conforms to the Intellectual Property guidelines specified in the NFC Forum's Intellectual Property Right Policy, as approved on November 9, 2004 and outlined in the NFC Forum Rules of Procedures, as approved on December 17, 2004.

1.8 Acronyms

This table defines all relevant terms and acronyms used in this specification.

Table 1. Acronyms

Acronyms	Definition
LSB	Least Significant Bit
NDEF	NFC Data Exchange Format
RFU	Reserved for Future Use
RTD	Record Type Description
URI	Uniform Resource Identifier
URL	Uniform Resource Locator (this is a special case of an URI)

2 Text Record

2.1 Introduction

The “Text” record contains freeform plain text. It can be used to describe a service or the contents of the tag, for example.

The Text record MAY appear as a sole record in an NDEF message [NDEF], but in this case the behavior is undefined and left to the application to handle. Typically, the Text record should be used in conjunction with other records to provide explanatory text.

2.2 Dependencies

There are no dependencies for the Text element.

2.3 Security Considerations

It is possible to write different text on the Text record than what the tag actually does, and thus spoof the user into doing something else than what he actually wanted (i.e., *phishing*). Thus it is a good idea for the user interface to use the Text field only as an informative field.

3 NDEF structure

3.1 Messaging Sequence

There is no particular messaging sequence available for this RTD.

3.2 Records Mapping

3.2.1 Syntax

The NFC Forum Well Known Type [NDEF], [NFC RTD] for the Text record is “T” (in NFC binary encoding: 0x54).

The data content is as follows:

Table 2. Text Record Content Syntax

Offset (bytes)	Length (bytes)	Content
0	1	Status byte. See Table 3.
1	<n>	ISO/IANA language code. Examples: “fi”, “en-US”, “fr-CA”, “jp”. The encoding is US-ASCII.
n+1	<m>	The actual text. Encoding is either UTF-8 or UTF-16, depending on the status bit.

The Status bit encodings are as described in Table 3. Any value marked RFU SHALL be ignored, and any software writing these bits SHALL use the value zero for these bits.

Table 3. Status Byte Encodings

Bit number (0 is LSB)	Content
7	0: The text is encoded in UTF-8 1: The text is encoded in UTF16
6	RFU (MUST be set to zero)
5..0	The length of the IANA language code.

The contents of the text field MAY be shown to the user. If multiple 'T' records exist, the one with the closest matching language to the user preference SHOULD be displayed. To have multiple text elements within a single application, context with the same language code SHOULD be considered an error.

Control characters (0x00-0x1F in UTF-8) should be removed prior to display, except for newline, line feed (0x0D, 0x0A) and tab (0x08) characters. Markup MUST NOT be embedded (please use the “text/xhtml” or other suitable MIME types). The Text record should be considered to be equal to the MIME type “text/plain; format=fixed”.

Line breaks in the text MUST be represented using the CRLF (so-called DOS convention, the sequence 0x0D,0x0A in UTF-8). The device may deal with the tab character as it wishes.

White space other than newline and tab SHOULD be collapsed, i.e., multiple space characters are to be considered a single space character.

To find the length of the actual text in bytes, you calculate the length via “m=(length of the payload – length of the IANA language code – 1)”

3.2.2 Structure

If the Text record describes an element, it SHOULD occur in the NDEF record list before the element it is describing. This makes it faster to find and display to the user if the element is very large.

3.3 Language Codes

All language codes MUST be done according to RFC 3066 [RFC3066]. The language code MAY NOT be omitted.

The language code length is encoded in the six least significant bits of the status byte. Thus it is easy to find by masking the status byte with the value 0x3F.

The language code is typically either two characters or five characters, though in the future, it is likely that it will be possible to have longer codes. At this time, IETF is considering an extension to RFC 3066 which will cover language codes up to 33 bytes in length [RFC 3066bis]. The two-character version disregards any dialects, and thus is used most often; for example, “fi” for Finnish, “jp” for Japanese, “fr” for French. However, in some cases you might want to differentiate between variants of the same language, such as providing US-English and British English versions via “en-US” and “en-UK” respectively.

3.4 UTF-16 Byte Order

The Unicode Byte-Order-Mark (BOM) in the actual string MUST be tolerated (i.e. no error condition). When generating a Text record, the BOM MAY be omitted. If the BOM is omitted, the byte order shall be big-endian (UTF-16 BE).

A. Example UTF-8 Encoding

Here's an example on how the English phrase "Hello, world!" would be encoded in UTF-8:

Table 4. Example: "Hello, world!"

Offset	Content	Explanation	Syntactical info
0	N/A	IL flag = 0 (no ID field), SF=1 (Short format)	NDEF record header
1	0x01	Length of the record name	
2	0x10	The length of the payload data (16 bytes)	
3	"T"	The binary encoding of the name, as defined in [1]	
4	0x02	Status byte: This is UTF-8, and has a two-byte language code	Payload
5	"en"	"en" is the ISO code for "English"	
7	"Hello, world!"	UTF-8 string "Hello, world!"	The actual body text.

B. Revision History

The following table outlines the revision history of the Text RTD Technical Specification.

Table 5. Revision History

Document Name	Revision and Release Date	Status	Change Notice	Supersedes
NFCForum-TS-RTD_Text_1.0	1.0, July 2006		None	First Revision

URI Record Type Definition

Technical Specification

NFC Forum™

RTD-URI 1.0

NFCForum-TS-RTD_URI_1.0

2006-07-24

NFC Forum, Inc.
401 Edgewater Place, Suite 600
Wakefield, MA, USA 01880

Contents

Tables

1 Overview

The URI Service RTD (Record Type Description) is an NFC RTD describing a record to be used with the NFC Data Exchange Format (NDEF) to retrieve a URI stored in a NFC-compliant tag or to transport a URI from one NFC device to another.

The URI (either a URN or URL) also provides a way to store URIs inside other NFC elements, such as a Smart Poster (please see the Smart Poster RTD for more information).

1.1 Objectives

The RTD defines the use of NDEF by the means of the NDEF records mapping.

1.2 Purpose

1.2.1 Mission Statement and Goals

The purpose of the URI RTD is to provide a "primitive" to contain URIs as defined by RFC 3986 in a compact manner.

1.3 References

[NDEF] "NFC Data Exchange Format Specification", NFC Forum, 2006.

[NFC RTD] "NFC Record Type Definition (RTD) Specification", NFC Forum, 2006.

[RFC 2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Harvard University, March 1997. http://www.apps.ietf.org/rfc/rfc2119.html

[RFC 3492] A. Costello: "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, March 2003. http://www.apps.ietf.org/rfc/rfc3492.html

[RFC 3986] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 3986, MIT/LCS, U.C. Irvine, Xerox Corporation, January 2005. http://www.apps.ietf.org/rfc/rfc3986.html

[RFC 3987] M. Duerst, M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, Microsoft Corporation, January 2005. http://rfc.net/rfc3987.html

[SMARTPOSTER] "Smart Poster RTD Specification", NFC Forum, 2006.

[URI SCHEME] List of Uniform Resource Identifier (URI) schemes registered by IANA. http://www.iana.org/assignments/uri-schemes

1.4 Administration

The URI RTD Specification is an open specification supported by the Near Field Communication Forum, Inc., located at:

401 Edgewater Place, Suite 600
Wakefield, MA, 01880

Tel.: +1 781-876-8955
Fax: +1 781-224-1239

http://www.nfc-forum.org

The Reference Applications technical working group maintains this specification.

This specification has been contributed to by Sony, Panasonic, Philips and Nokia.

1.5 Special Word Usage

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119.

1.6 Name and Logo Usage

The Near Field Communication Forum’s policy regarding the use of the trademarks *NFC Forum* and the NFC Forum logo is as follows:

- Any company MAY claim compatibility with NFC Forum specifications, whether a member of the NFC Forum or not.
- Permission to use the NFC Forum logos is automatically granted to designated members only as stipulated on the most recent Membership Privileges document, during the period of time for which their membership dues are paid.
- Member’s distributors and sales representatives MAY use the NFC Forum logo in promoting member’s products sold under the name of the member.
- The logo SHALL be printed in black or in color as illustrated on the Logo Page that is available from the NFC Forum at the address above. The aspect ratio of the logo SHALL be maintained, but the size MAY be varied. Nothing MAY be added to or deleted from the logos.
- Since the NFC Forum name is a trademark of the Near Field Communication Forum, the following statement SHALL be included in all published literature and advertising material in which the name or logo appears:

  NFC Forum and the NFC Forum logo are trademarks of the Near Field Communication Forum.

1.7 Intellectual Property

The URI Record Type Definition Specification conforms to the Intellectual Property guidelines specified in the NFC Forum's Intellectual Property Right Policy, as approved on November 9, 2004 and outlined in the NFC Forum Rules of Procedures, as approved on December 17, 2004.

1.8 Acronyms

This table defines all relevant terms and acronyms used in this specification.

Table 1. Acronyms

Acronyms	Definition
NDEF	NFC Data Exchange Format
URI	Uniform Resource Identifier
URL	Uniform Resource Locator (this is a special case of an URI)
RFU	Reserved for Future Use
NFC	Near Field Communication

2 URI Service

This document defines URI Service with data model, describing the application scenarios for simple Smart Poster applications, the structure of an URI located on an NFC compliant device or tag, and provides examples.

The URI record type MAY also be used as a part of some other RTD, in which case it implies no specific action. A typical example of this might be a case where the developer wants to build his own record type containing multiple URLs. In this case, it is impossible to divine the meaning of each URL automatically, so it is left to the handler taking care of the developer's own type.

Devices are NOT required to implement any particular URI protocol.

2.1 NDEF Message Sequences

There are no specific message sequences.

2.2 Dependencies

The Smart Poster RTD [SMARTPOSTER] may be considered to be an extended version of the URI RTD. It uses auxiliary records to add metadata to the URI.

3 NDEF Structure

3.1 Messaging Sequence

There is no particular messaging sequence.

3.2 Records Mapping

3.2.1 URI Record Type

The Well Known Type for an URI record is “U” (0x55 in the NDEF binary representation).

The structure of an URI record is described below.

Table 2. URI Record Contents

Name	Offset	Size	Value	Description
Identifier code	0	1 byte	URI identifier code	The URI identifier code, as specified in Table 3.
URI field	1	N	UTF-8 string	The rest of the URI, or the entire URI (if identifier code is 0x00).

3.2.2 URI Identifier Code

In order to shorten the URI, the first byte of the record data describes the protocol field of an URI. The following table MUST be used to encode and decode the URI, though applications MAY use the 0x00 value to denote no prefixing when encoding, regardless of whether there actually is a suitable abbreviation code.

For explanations of the different protocols, please refer to the protocol documentations themselves. NFC devices are not required to support any particular protocol.

Table 3. Abbreviation Table

Decimal	Hex	Protocol
0	0x00	N/A. No prepending is done, and the URI field contains the unabridged URI.
1	0x01	http://www.
2	0x02	https://www.
3	0x03	http://
4	0x04	https://
5	0x05	tel:
6	0x06	mailto:
7	0x07	ftp://anonymous:anonymous@
8	0x08	ftp://ftp.
9	0x09	ftps://

Decimal	Hex	Protocol
10	0x0A	sftp://
11	0x0B	smb://
12	0x0C	nfs://
13	0x0D	ftp://
14	0x0E	dav://
15	0x0F	news:
16	0x10	telnet://
17	0x11	imap:
18	0x12	rtsp://
19	0x13	urn:
20	0x14	pop:
21	0x15	sip:
22	0x16	sips:
23	0x17	tftp:
24	0x18	btspp://
25	0x19	btl2cap://
26	0x1A	btgoep://
27	0x1B	tcpobex://
28	0x1C	irdaobex://
29	0x1D	file://
30	0x1E	urn:epc:id:
31	0x1F	urn:epc:tag:
32	0x20	urn:epc:pat:
33	0x21	urn:epc:raw:
34	0x22	urn:epc:
35	0x23	urn:nfc:
36…255	0x24..0xFF	RFU

For example, if the content of this field is 0x02, and the content of the URI field reads as “nfc-forum.org”, the resulting URI is “https://www.nfc-forum.org”.

If the content this field is zero (0x00), then NO prepending SHALL be done.

All fields marked RFU SHALL be treated as if they were value zero (no prepending). A compliant system MUST NOT produce values that are marked RFU.

3.2.3 URI Field

This field provides the URI as per RFC 3987 [RFC 3987] (so that it is actually an IRI, or Internationalized Resource Identifier, but for legacy reasons we use the word URI). This IRI can be a URL or URN as explained before. The encoding used MUST be UTF-8, unless the URI scheme specifies some particular encoding.

The length of the IRI can be calculated by taking the length of the payload, and subtracting 1 for the protocol abbreviation code byte. This is the length in bytes, not in characters (as UTF-8 characters can occupy more than one byte).

URIs are defined only in the 7-bit US-ASCII space. Therefore, a compliant application SHOULD transform the UTF-8 IRI string to a 7-bit US-ASCII string by changing code points above 127 into the proper encoding. This coding has been defined in the RFC 3987 [RFC 3987] and IDN [RFC 3492] documents. For different schemes, the encoding may be different.

For example, if the URI (after the prepending of the URI type field) contains the following string: "http://www.hääyö.com/", it is transformed, as per standard IDN [RFC 3492] rules, into "http://www.xn--hy-viaa5g.com" before acting on it. Most modern applications already support this new Internationalized Resource Identifier (IRI) scheme. It is RECOMMENDED that implementations include support for IRI where display of the URI in human-readable form is anticipated.

To clarify: yes, the URI MAY contain UTF-8 characters. However, the Internet cannot handle them, and therefore the URI needs to be transformed before use. For most devices, this conversion is handled by the application.

Any character value within the URI between (and including) 0 and 31 SHALL be recorded as an error, and the URI record to be discarded. Any invalid UTF-8 sequence SHALL be considered an error, and the entire URI record SHALL be discarded.

4 Handling Guideline

The URI RTD does not define any specific action that the device is required to perform. This is left to the implementation.

Please see the Smart Poster RTD [SMARTPOSTER] for an example on how to use the URI RTD in your own application.

A. Examples

These examples omit the MB and ME flags from the URI RTD, and assume the Short Record format. See the NDEF specification [NDEF] for more information.

A.1 Simple URL with No Substitution

To put the URL http://www.nfc.com on a tag using the NDEF protocol, add the following byte sequence. Total length: 12 bytes.

Table 4. Simple URL with No Substitution

Offset	Content	Explanation
0	0xD1	SR = 1, TNF = 0x01 (NFC Forum Well Known Type), ME=1, MB=1
1	0x01	Length of the Record Type (1 byte)
2	0x08	Length of the payload (8 bytes)
3	0x55	The URI record type ("U")
4	0x01	URI identifier ("http://www.")
5	0x6e 0x66 0x63 0x2e 0x63 0x6f 0x6d	The string "nfc.com" in UTF-8.

A.2 Storing a Telephone Number

To store a telephone number (for example, to make a mobile phone make a call to this number), use the following byte sequence. The number is '+358-9-1234567'. Total length is 17 bytes.

Table 5. Storing a Telephone Number

Offset	Content	Explanation
0	0xD1	SR = 1, TNF = 0x01 (NFC Forum Well Known Type), MB=1, ME=1
1	0x01	Length of the Record Type (1 byte)
2	0x0D	Length of the payload (13 bytes)
3	0x55	The Record Name ("U")
4	0x05	Abbreviation for "tel:"
5	0x2b 0x33 0x35 0x38 0x39 0x31 0x32 0x33 0x34 0x35 0x36 0x37	The string "+35891234567" in UTF-8.

A.3 Storing a Proprietary URI on the Tag

To store a proprietary URI, you can use the following byte sequence. The URI in this case is “mms://example.com/download.wmv”. Total length is 35 bytes.

Table 6. Storing a Proprietary URI on the Tag

Offset	Content	Explanation
0	0xD1	SR = 1, TNF = 0x01 (NFC Forum Well Known Type), MB=1, ME=1
1	0x01	Length of the Record Type (1 byte)
2	0x1F	Length of the payload (31 bytes)
3	0x55	The Record Name (“U”)
4	0x00	No abbreviation
5	0x6d 0x6d 0x73 0x3a 0x2f 0x2f 0x65 0x78 0x61 0x6d 0x70 0x6c 0x65 0x2e 0x63 0x6f 0x6d 0x2f 0x64 0x6f 0x77 0x6e 0x6c 0x6f 0x61 0x64 0x2e 0x77 0x6d 0x76	The string “mms://example.com/download.wmv“.

B. Revision History

The following table outlines the revision history of the RTD_URI Technical Specification.

Table 7. Revision History

Document Name	Revision and Release Date	Status	Change Notice	Supersedes
NFCForum-TS-RTD_URI_1.0	1.0, July 2006	Final	None	