NFC Data Exchange Format

Technical Specification

Version 1.0

2021-08-23

[NDEF]

NFC Forum™

Contents

Figures

Tables

Requirements

1 Overview

The NFC Data Exchange Format (NDEF) specification defines a message encapsulation format to exchange information between two NFC Forum Devices.

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 can 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 NDEF Payload Length is an unsigned integer that indicates the number of octets in the payload. A compact, NDEF short-record layout is provided for very small payloads.

The optional NDEF Payload Identifier enables association of multiple payloads and cross-referencing between them.

NDEF Payloads can 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.

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 NDEF Record Types in detail — NDEF 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.

An NFC Forum Device can process the NDEF information independently of the way it has received the NDEF Message.

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 stream-oriented protocols like TCP are used.

1.2 Applicable Documents or References

[NFC RTD]	NFC Record Type Definition (RTD) Specification, NFC Forum
[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.
[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.
[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 NFC 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: +1 781-610-9864

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

1.4 Trademark and Logo Usage

The Near Field Communication Forum's policy regarding the use of trademarks and logos is described in the NFC Forum Brand Identity Guidelines and N-Mark Usage Guidelines, which can be found on the NFC Forum website.

1.5 Intellectual Property

This document conforms to the Intellectual Property guidelines specified in the NFC Forum *Intellectual Property Rights Policy*, as outlined in the NFC Forum *Rules of Procedure*. These documents are available on the NFC Forum website.

1.6 Special Word Usage

The key words "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT" and "MAY" in this document are to be interpreted as described in [RFC 2119].

1.7 Abbreviations

Table 1 contains the definitions of the abbreviations and acronyms used in this document.

Table 1: Abbreviations

Abbreviation	Description
CF	Chunk Flag
NDEF	NFC Data Exchange Format
QoS	Quality of Service
RFU	Reserved for Future Use
URI	Uniform Resource Identifier

1.8 Glossary

Big Endian

A method of recording or transmitting numerical data of more than one byte, with the most significant byte placed at the beginning.

Chunked Payload

Application data that has been partitioned into multiple chunks, each carried in a separate NDEF Record.

NDEF Application

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

NDEF Generator

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

NDEF Message

The basic message construct defined by this specification. An NDEF Message contains one or more NDEF Records.

NDEF Parser

An entity or module that parses NDEF Messages and hands off the payloads to an NDEF Application.

NDEF Payload

The application data carried within an NDEF Record.

NDEF Payload Identifier

An optional Uniform Resource Identifier (URI) that can be used to identify a payload.

NDEF Payload Length

An unsigned integer that indicates the number of octets in the payload of a single NDEF Record.

NDEF Payload Type

An identifier that indicates the type of the payload.

NDEF Record

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

NDEF Record Chunk

An NDEF Record that contains a chunk of a payload rather than a full payload.

NDEF Short Record

An NDEF Record that allows payloads or chunks of up to 255 bytes to be carried.

NFC Forum Device

A device that supports at least one communication protocol for at least one communication mode defined by the NFC Forum specifications. Currently the following NFC Forum Devices are defined:
NFC Universal Device, NFC Tag Device and NFC Reader Device.

NFC Reader Device

An NFC Forum Device that supports the following Modus Operandi: Reader/Writer. It can also support Initiator.

NFC Tag Device

An NFC Forum Device that supports at least one communication protocol for Card Emulator and NDEF.

NFC Universal Device

An NFC Forum Device that supports the following Modus Operandi: Initiator, Target, and Reader/Writer. It can also support Card Emulator.

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 232-1 octets in size. NDEF Records can be chained together to support larger payloads. An NDEF Record carries three parameters for describing its payload: the NDEF Payload Length, the NDEF Payload Type, and an optional NDEF Payload Identifier. The purpose of these parameters is as follows:

NDEF Payload Length

> The NDEF Payload Length indicates the number of octets in the payload (see Section 2.4.1). Providing the NDEF Payload Length within the first 8 octets of an NDEF Record makes efficient record boundary detection possible.

NDEF Payload Type

> The NDEF Payload Type identifier indicates the type of the payload. NDEF supports Uniform Resource Identifiers (URIs) [RFC 3986], MIME media type constructs [RFC 2046], and an NFC-specific type format as type identifiers (see Section 2.4.2). The indication of the type of the payload makes it possible to dispatch the payload to the appropriate NDEF Application.

NDEF Payload Identifier

> A payload can 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: an NDEF Application is designed 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 via NFC to another NFC Forum Device where it is received and parsed. The NDEF Parser deconstructs the NDEF Message and hands the payloads to a (potentially different) NDEF Application. Each NDEF Message has to 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 NDEF Applications involved to provide any additional functionality such as QoS that they need as part of the overall system in which they participate.

Requirements 1: Intended Usage

NFC Forum Device	
2.2.1.1	Each NDEF Message SHALL be sent or received in its entirety.

2.3 NDEF Encapsulation Constructs

2.3.1 NDEF Message

An NDEF Message is composed of one or more NDEF Records. The first NDEF Record in an NDEF Message is marked with the MB (Message Begin) flag set and the last NDEF Record in the NDEF Message is marked with the ME (Message End) flag set (see Sections 2.6.2 and 2.6.3). The minimum NDEF Message length is one NDEF Record which is achieved by setting both the MB and the ME flag in the same NDEF 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 cannot overlap; that is, the MB and the ME flags cannot be used to nest NDEF Messages. NDEF Messages can 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 NDEF Records

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

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

Requirements 2: NDEF Encapsulation - Message

NDEF Message Encapsulation	
2.3.1.1	An NDEF Message SHALL be composed of one or more NDEF Records.
2.3.1.2	The first NDEF Record in an NDEF Message SHALL be marked with the MB (Message Begin) flag set.
2.3.1.3	The last NDEF Record in the NDEF Message SHALL be marked with the ME (Message End) flag set.
2.3.1.4	NDEF Messages SHALL NOT overlap; that is, the MB and the ME flags SHALL NOT be used to nest NDEF Messages.
2.3.1.5	NDEF Messages MAY be nested by carrying a full NDEF Message as a payload within an NDEF Record.

2.3.2 NDEF Record

An NDEF 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 NDEF Record Chunks

An NDEF 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 NDEF Record Chunks serialized within the same NDEF Message.

Chunking is not a mechanism for introducing multiplexing or data streaming into NDEF and it is forbidden to be used for those purposes. Chunking is a mechanism to reduce the need for outbound buffering on the generating side, 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 NDEF Record Chunk followed by zero or more middle NDEF Record Chunks and finally by a terminating NDEF Record Chunk. Each NDEF Record Chunk is encoded as an NDEF Record that uses the following encoding rules:

- The initial NDEF Record Chunk is an NDEF Record with the CF (Chunk Flag) set (see Section 2.6.4). The type of the entire Chunked Payload is 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 NDEF Record indicates the size of the data carried in the PAYLOAD field of the initial NDEF Record only, not the entire payload size (see Section 2.4.1).
- Each middle NDEF Record Chunk is an NDEF Record with its CF set, indicating that this NDEF Record Chunk contains the next chunk of data of the same type and with the same identifier as the initial NDEF Record Chunk. The values of the TYPE_LENGTH and the IL fields are zero, and the TNF (Type Name Format) field value is 0x06 (Unchanged) (see Section 2.6.7). The PAYLOAD_LENGTH field indicates the size of the data carried in the PAYLOAD field of only this single middle NDEF Record (see Section 2.4.1).
- The terminating NDEF Record Chunk is an NDEF Record with its CF cleared, indicating that this NDEF Record Chunk contains the last chunk of data of the same type and with the same

identifier as the initial NDEF Record Chunk. As in the middle NDEF Record Chunks, the values of the TYPE_LENGTH and the IL fields are zero, and the TNF (Type Name Format) field value is 0x06 (Unchanged) (see Section 2.6.7). The PAYLOAD_LENGTH field indicates the size of the data carried in the PAYLOAD field of this terminating NDEF Record Chunk (see Section 2.4.1).

A Chunked Payload has to be entirely encapsulated within a single NDEF Message. That is, a Chunked Payload cannot span multiple NDEF Messages. As a consequence, neither an initial nor a middle NDEF Record Chunk can have the ME (Message End) flag set.

Requirements 3: NDEF Encapsulation – NDEF Record Chunks

Record Chunks Encapsulation	
2.3.3.1	Chunking SHALL NOT be used for introducing multiplexing or data streaming into NDEF.
2.3.3.2	Each Chunked Payload SHALL be encoded as an initial NDEF Record Chunk followed by 0 or more middle NDEF Record Chunks and finally by a terminating NDEF Record Chunk.
2.3.3.3	The initial NDEF Record Chunk SHALL be an NDEF record with the CF (Chunk Flag) set.
2.3.3.4	The type of the entire Chunked Payload SHALL be indicated in the TYPE field of the initial NDEF Record Chunk.
2.3.3.5	The PAYLOAD_LENGTH field of the initial NDEF record indicates the size of the data carried in the PAYLOAD field of the initial NDEF record only, not the entire payload size.
2.3.3.6	Each middle NDEF Record Chunk SHALL be an NDEF record with the CF set.
2.3.3.7	For each middle NDEF Record Chunk the value of the TYPE_LENGTH and the IL fields SHALL be 0.
2.3.3.8	For each middle NDEF Record Chunk the TNF (Type Name Format) field value SHALL be 0x06 (Unchanged).
2.3.3.9	For each middle NDEF Record Chunk the PAYLOAD_LENGTH field SHALL indicate the size of the data carried in the PAYLOAD field of this single NDEF Record only.
2.3.3.10	The terminating NDEF Record Chunk SHALL be an NDEF record with the CF cleared.
2.3.3.11	For the terminating NDEF Record Chunk the value of the TYPE_LENGTH and the IL fields SHALL be 0.
2.3.3.12	For the terminating NDEF Record Chunk the TNF (Type Name Format) field value SHALL be 0x06 (Unchanged).
2.3.3.13	For the terminating NDEF Record Chunk the PAYLOAD_LENGTH field SHALL indicate the size of the data carried in the PAYLOAD field of this NDEF Record only.
2.3.3.14	A Chunked Payload SHALL be entirely encapsulated within a single NDEF Message.
2.3.3.15	An initial NDEF Record Chunk SHALL NOT have the ME (Message End) flag set.
2.3.3.16	A middle NDEF Record Chunk SHALL NOT have the ME (Message End) flag set.

2.4 NDEF Payload Description

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

2.4.1 NDEF Payload Length

Regardless of the relationship of an NDEF Record to other NDEF Records, the NDEF Payload Length always indicates the length of the payload encapsulated in this NDEF Record. The length of the payload is indicated in the PAYLOAD_LENGTH field. The PAYLOAD_LENGTH field is one octet for NDEF Short Records and four octets for normal records. NDEF Short Records are indicated by setting the SR bit flag to a value of 1 (see Section 2.6.5). Zero is a valid NDEF Payload Length.

Requirements 4: NDEF Payload Length

NDEF Payload Length	
2.4.1.1	The PAYLOAD_LENGTH field SHALL be four octets for normal NDEF Records, representing a 32-bit unsigned integer.
2.4.1.2	The PAYLOAD_LENGTH field of a normal NDEF Record SHALL have a value between 0 and 2^{32}-1.
2.4.1.3	The PAYLOAD_LENGTH field SHALL be one octet for records with an SR (Short Record) bit flag value of 1, representing an 8-bit unsigned integer.
2.4.1.4	The PAYLOAD_LENGTH field of an NDEF Short Record SHALL have a value between 0 and 255.

2.4.2 NDEF Payload Type

The NDEF Payload Type of an NDEF Record indicates the kind of data being carried in the payload of that NDEF Record. This can be used to guide the processing of the payload at the discretion of the NDEF Application. The type of the first NDEF Record, by convention, ought to provide the processing context not only for the first NDEF Record but for the whole NDEF Message. Additional context for processing the NDEF Message can be provided, for example, by the link layer service access point (LSAP) or transport service port (e.g. TCP, UDP, etc.) at which the NDEF 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 NDEF Payload Types is entirely at the discretion of the NDEF 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 by using the TNF (Type Name Format) field (see Section 2.6.7). This specification supports TYPE field values in the forms 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 NDEF Payload Types that support NFC Forum reference applications [NFC RTD]; URIs provide for decentralized control of the value space; and 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]. Use of non-registered media types is discouraged. The URI scheme registration process is described in [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. NDEF Records that carry a payload with an XML-based message type can use the XML namespace identifier of the root element as the TYPE field value. A SOAP/1.1 message, for example, can 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 cannot 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.

NDEF Records that carry a payload with an existing, registered media type ought to 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]. Similarly, the media type

message/http

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

application/xml; charset="utf-16"

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

Requirements 5: NDEF Payload Type

Payload Type	
2.4.2.1	The type of the first NDEF Record SHOULD provide the processing context not only for the first NDEF Record but for the whole NDEF Message.
2.4.2.2	For URI types, only well known URI schemes registered by IANA SHOULD be used.
2.4.2.3	NDEF 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.
2.4.2.4	NDEF Records that carry a payload with an existing, registered media type SHOULD carry a TYPE field value of that media type.

2.4.3 Payload Identification

The optional NDEF Payload Identifier allows NDEF Applications to identify the payload carried within an NDEF Record. By providing an NDEF 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 NDEF Application to define in the language it prefers.

It is important that NDEF Payload Identifiers are maintained so that references to those payloads are not broken. If NDEF 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 Data Transmission Order

The order of transmission of the NDEF Record described in this document is resolved to the octet level. For diagrams that show 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 that represents 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.

Requirements 6: Data Transmission Order

Data Transmission Order	
2.5.1.1	Quantities SHALL be transmitted in a Big Endian manner with the most significant octet transmitted first.

2.6 NDEF 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.

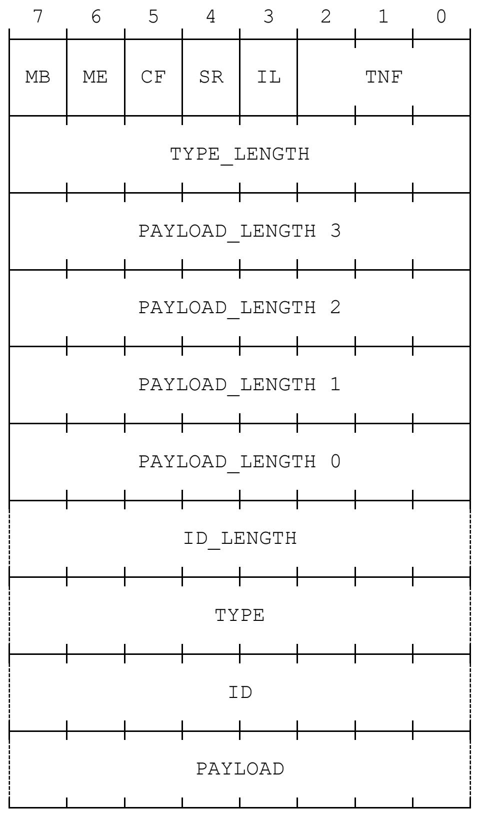

Figure 3. NDEF Record Layout

Requirements 7: Normal Record Layout

Normal Record Layout	
2.6.1.1	The fields of a normal NDEF Record SHALL be coded and interpreted as shown in Figure 3.

2.6.2 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).

2.6.3 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 a Chunked Payload the ME flag is set only in the terminating NDEF Record Chunk of that payload (see Section 2.3.3).

2.6.4 CF (Chunk Flag)

The CF is a 1-bit field indicating that this is either the first NDEF Record Chunk or a middle NDEF Record Chunk of a Chunked Payload (see Section 2.3.3 for a description of how to encode a Chunked Payload).

2.6.5 SR (Short Record)

The SR flag is a 1-bit field indicating, if set, that the PAYLOAD_LENGTH field is a single octet. This NDEF 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.

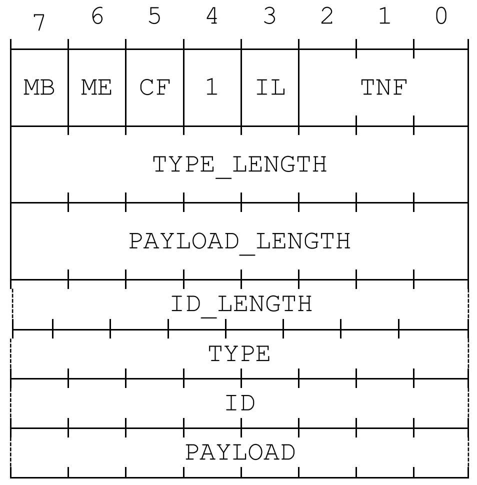

Figure 4. NDEF Short Record Layout (SR=1)

While it is tempting for implementers to choose one or the other record layout exclusively for a given application, NDEF Parsers accepts both normal and short record layouts. NDEF Generators MAY generate either record layout as they deem appropriate. A single NDEF Message can contain both normal and short records.

Requirements 8: SR (Short Record)

SR (Short Record)	
2.6.5.1	The fields of a short NDEF Record SHALL be coded and interpreted as shown in Figure 3.
2.6.5.2	NDEF Parsers SHALL accept both normal and NDEF Short Record layouts.
2.6.5.3	NDEF Parsers SHALL accept single NDEF Messages composed of both normal and short records.

2.6.6 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 NDEF Record header and the ID field is also omitted from the NDEF Record.

Requirements 9: IL (ID_LENGTH Field is Present)

IL (ID_LENGTH field is present)	
2.6.6.1	If the IL flag is 1, the ID_LENGTH field SHALL be present.
2.6.6.2	If the IL flag is 0, the ID_LENGTH field and the ID field SHALL NOT be present.

2.6.7 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 2. TNF Field Values

Value	Description
0x00	Empty
0x01	NFC Forum Well Known Type [NFC RTD]
0x02	Media-type as defined in [RFC 2046]
0x03	Absolute URI as defined in [RFC 3986]
0x04	NFC Forum External Type [NFC RTD]
0x05	Unknown
0x06	Unchanged (see Section 2.3.3)
All other values	RFU

The value 0x00 (Empty) indicates that there is no type, ID, or payload associated with this NDEF Record. This TNF value can be used whenever an empty NDEF Record is needed – for example, to terminate an NDEF Message when there is no payload defined by the NDEF 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].

The value 0x03 (absolute-URI) indicates that the TYPE field contains a value that follows the absolute-URI BNF construct defined by [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) is 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 it is used, the TYPE_LENGTH field is zero and the TYPE field is omitted from the NDEF Record. Regarding implementation, it is recommended that an NDEF Parser that receives an NDEF Record of this type, without further context to its use, provide a mechanism for storing but not processing the payload (see Section 3.2).

The value 0x06 (Unchanged) is used in all middle NDEF Record Chunks and the terminating NDEF Record Chunk used in Chunked Payloads (see Section 2.3.3). It is not used in any other NDEF Record. When used, the TYPE_LENGTH field is zero and the TYPE field is omitted from the NDEF Record.

There is no default value for the TNF field. RFU TNF field values are for future use and cannot be used. An NDEF Parser that receives an NDEF Record with an unknown or unsupported TNF field value ought to treat it as 0x05 (Unknown).

Requirements 10: TNF (Type Name Format)

TNF (Type Name Format)	
2.6.7.1	The TNF field SHALL be coded and interpreted as defined in Table 2.
2.6.7.2	An NDEF Generator SHALL set the IL flag to 0b and the TYPE_LENGTH and PAYLOAD_LENGTH fields to zero. As a consequence, the ID_LENGTH, TYPE, ID and PAYLOAD fields are omitted from the NDEF Record. An NDEF Parser SHALL accept this as an empty NDEF Record also if IL is set to 1b and ID_LENGTH is zero. An NDEF Parser that receives an NDEF Record of this type, without further context to its use, SHOULD provide a mechanism for storing but not processing the payload
2.6.7.3	If the TNF value is 0x01, the content of the PAYLOAD field SHALL comply with the definitions for NFC Forum Well Known Types in [NFC RTD].
2.6.7.4	If the TNF value is 0x02, the content of the PAYLOAD field SHALL comply with the definitions for media-types in [RFC 2046].
2.6.7.5	If the TNF value is 0x03, the content of the PAYLOAD field SHALL comply with the definitions for media-types in [RFC 3986].
2.6.7.6	If the TNF value is 0x04, the content of the PAYLOAD field SHALL comply with the definitions for NFC Forum External Types in [NFC RTD].
2.6.7.7	If the TNF value is 0x05 (Unknown), the TYPE_LENGTH field SHALL be 0 and the TYPE field SHALL be omitted from the NDEF Record.
2.6.7.8	If the TNF value is 0x06 (Unchanged), the TYPE_LENGTH field SHALL be 0 and the TYPE field SHALL be omitted from the NDEF Record.
2.6.7.9	The TNF field SHALL NOT contain an RFU value.
2.6.7.10	An NDEF Parser that receives an NDEF Record with an unknown or unsupported TNF field value SHOULD treat it as 0x05 (Unknown).

2.6.8 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 2.6.7).

2.6.9 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 NDEF Record header. An ID_LENGTH of zero octets is allowed and, in such cases, the ID field is omitted from the NDEF Record.

Requirements 11: ID_LENGTH

ID_LENGTH	
2.6.9.1	If the ID_LENGTH field has the value 0, the ID field SHALL NOT be present.

2.6.10 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 2.6.5).

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

If the SR flag is clear, the PAYLOAD_LENGTH field is four octets that represent a 32-bit unsigned integer. The transmission order of the octets is MSB-first (see Section 2.5).

An NDEF 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).

Requirements 12: PAYLOAD_LENGTH

PAYLOAD_LENGTH	
2.6.10.1	If the PAYLOAD_LENGTH field value is 0, the PAYLOAD field SHALL NOT be present.

2.6.11 TYPE

The value of the TYPE field is an identifier that describes the type of the payload (see Section 2.4.2). The value of the TYPE field follows the structure, encoding and format implied by the value of the TNF field (see Section 2.6.7).

An NDEF Parser that receives an NDEF Record with a TNF field value that it supports, but with an unknown TYPE field value, ought to interpret the type identifier of that NDEF Record as if the TNF field value were 0x05 (Unknown).

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

Requirements 13: TYPE

TYPE	
2.6.11.1	The value of the TYPE field SHALL follow the structure, encoding and format implied by the value of the TNF field.
2.6.11.2	An NDEF Parser that receives an NDEF Record with a TNF field value that it supports, but with an unknown TYPE field value, SHOULD interpret the type identifier of that NDEF Record as if the TNF field value were 0x05 (Unknown).
2.6.11.3	The type identifier SHOULD be globally unique and maintained with stable and well-defined semantics over time.

2.6.12 ID

The value of the ID field is an identifier in the form of an URI reference [RFC 3986] (see Sections 2.4.3 and 3.4). The required uniqueness of the NDEF 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 NDEF Applications that use relative URIs need to provide an actual or a virtual base URI (see [RFC 3986]).

Middle and *terminating* NDEF Record Chunks (that is, records that do not contain the *initial* chunk of a Chunked Payload; see Section 2.3.3) do not have an ID field. All other NDEF Records can have an ID field.

Requirements 14: ID

ID	
2.6.12.1	Middle and terminating NDEF Record Chunks SHALL NOT have an ID field. All other NDEF Records MAY have an ID field.
2.6.12.2	NDEF Applications that use relative URIs SHALL provide an actual or a virtual base URI (see [RFC 3986])

2.6.13 PAYLOAD

The PAYLOAD field carries the payload intended for the NDEF Application. Any internal structure of the data carried within the PAYLOAD field is opaque in NDEF.

3 Special considerations

3.1 Internationalization

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

3.2 Security

Implementers need to pay special attention to the security implications of any NDEF Record Types that can cause the remote execution of any actions in the recipient's environment. Before accepting NDEF Records of any type, an application needs to be aware of the particular security implications associated with that type.

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

3.3 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 sizes 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 NDEF Short Record layout (see Section 2.6.5). The maximum size of the 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 NDEF Applications will have the ability or the need to handle payload content, payload IDs or type identifiers of these maximal sizes. NDEF Parsers that are resource constrained can choose to reject messages that are not sized to fit their specific needs.

However, NDEF Parsers are not allowed to reject an NDEF Message based solely on the value of the SR flag.

Requirements 15: Maximum Field Sizes

Maximum Field Sizes	
3.3.1.1	NDEF Parsers SHALL NOT reject an NDEF Message based solely on the value of the SR flag.
3.3.1.2	NDEF Parsers MAY reject messages that include NDEF Records with TYPE, ID, or PAYLOAD fields larger than their design limits.

3.4 Use of URIs in NDEF

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

The use of IP addresses in URIs needs to be avoided whenever possible (see [RFC 1900]). However, when IP addresses are used, it is recommended that the implementation supports the literal format for IPv6 addresses in URIs, as described in [RFC 2732] .

NDEF does not define any equivalence rules for URIs in general, as these are defined by the individual URI schemes and by [RFC 3986]. However, because of inconsistencies in 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 in [RFC 3986].

Requirements 16: Use of URIs

Use of URIs	
3.4.1.1	The use of IP addresses in URIs SHOULD be avoided whenever possible (see [RFC 1900]). However, when used, the literal format for IPv6 addresses in URIs described by [RFC 2732] SHOULD be supported.
3.4.1.2	Generators of NDEF Messages SHOULD rely only on the most rudimentary equivalence rules defined by [RFC 3986].

A. Exhibit A

No items have been included in Exhibit A.

B. Revision History

Table 3 outlines the revision history of the NFC Data Exchange Format Technical Specification.

Table 3: Revision History

Document Name	Revision and Release Date	Status	Change Notice	Supersedes
NFC Data Exchange Format	1.0, July 2006	Final	None	
NFC Data Exchange Format	1.0, November 2017	Final	Editorial update of format, style and terminology.	Version 1.0, July 2006
NFC Data Exchange Format	Version 1.0 February 2020	Final	Editorial update, including revision of license page.	Version 1.0, November 2017
NFC Data Exchange Format	Version 1.0 August 2021	Final	Editorial update, - Clarification for empty NDEF Message - Alignment of field names	Version 1.0, February 2020