



Logical Link Control Protocol

Technical Specification

NFC Forum™

LLCP 1.0

NFCForum-TS-LLCP_1.0

2009-12-11

RESTRICTIONS ON USE

This specification is copyright © 2005-2009 by the NFC Forum, and was made available pursuant to a license agreement entered into between the recipient (Licensee) and NFC Forum, Inc. (Licensor) and may be used only by Licensee, and in compliance with the terms of that license agreement (License). If you are not the Licensee, you are not authorized to make any use of this specification. However, you may obtain a copy at the following page of Licensor's Website: http://www.nfc-forum.org/specs/spec_license after entering into and agreeing to such license terms as Licensor is then requiring. On the date that this specification was downloaded by Licensee, those terms were as follows:

1. LICENSE GRANT.

Licensor hereby grants Licensee the right, without charge, to copy (for internal purposes only) and share the Specification with Licensee's members, employees and consultants (as appropriate). This license grant does not include the right to sublicense, modify or create derivative works based upon the Specification.

2. NO WARRANTIES.

THE SPECIFICATION IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY, COMPLETENESS AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL LICENSOR, ITS MEMBERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE SPECIFICATION.

3. THIRD PARTY RIGHTS.

Without limiting the generality of Section 2 above, LICENSOR ASSUMES NO RESPONSIBILITY TO COMPILE, CONFIRM, UPDATE OR MAKE PUBLIC ANY THIRD PARTY ASSERTIONS OF PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS THAT MIGHT NOW OR IN THE FUTURE BE INFRINGED BY AN IMPLEMENTATION OF THE SPECIFICATION IN ITS CURRENT, OR IN ANY FUTURE FORM. IF ANY SUCH RIGHTS ARE DESCRIBED ON THE SPECIFICATION, LICENSOR TAKES NO POSITION AS TO THE VALIDITY OR INVALIDITY OF SUCH ASSERTIONS, OR THAT ALL SUCH ASSERTIONS THAT HAVE OR MAY BE MADE ARE SO LISTED.

4. TERMINATION OF LICENSE.

In the event of a breach of this Agreement by Licensee or any of its employees or members, Licensor shall give Licensee written notice and an opportunity to cure. If the breach is not cured within thirty (30) days after written notice, or if the breach is of a nature that cannot be cured, then Licensor may immediately or thereafter terminate the licenses granted in this Agreement.

5. MISCELLANEOUS.

All notices required under this Agreement shall be in writing, and shall be deemed effective five days from deposit in the mails. Notices and correspondence to the NFC Forum address as it appears below. This Agreement shall be construed and interpreted under the internal laws of the United States and the Commonwealth of Massachusetts, without giving effect to its principles of conflict of law.

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

Contents

1	Overview	4
1.1	Features and Scope	4
1.2	Applicable Documents or References	5
1.2.1	Normative Authorities	5
1.2.2	Normative References	6
1.2.3	Informative References	6
1.3	Administration	6
1.4	Special Word Usage	6
1.5	Name and Logo Usage	6
1.6	Intellectual Property	7
1.7	Glossary	7
1.8	Acronyms and Abbreviations	8
2	Definitions	10
2.1	Bit Significance in Diagrams and Tables	10
2.2	LLC Protocol Version	10
3	Architecture	11
3.1	Components	11
3.2	LLCP Data Transport Services	12
3.2.1	Link Service Class	12
3.2.2	Connectionless Transport	12
3.2.3	Connection-oriented Transport	12
4	LLC Elements	14
4.1	LLC PDU Format	14
4.2	Elements of the LLC PDU	14
4.2.1	Address Fields	14
4.2.2	PDU Type	15
4.2.3	Sequence Field	16
4.2.4	Information Field	16
4.3	PDU Descriptions	16
4.3.1	Symmetry (SYMM)	16
4.3.2	Parameter Exchange (PAX)	17
4.3.3	Aggregated Frame (AGF)	17
4.3.4	Unnumbered Information (UI)	18
4.3.5	Connect (CONNECT)	19
4.3.6	Disconnect (DISC)	19
4.3.7	Connection Complete (CC)	19
4.3.8	Disconnected Mode (DM)	20
4.3.9	Frame Reject (FRMR)	21
4.3.10	Information (I)	22
4.3.11	Receive Ready (RR)	23
4.3.12	Receive Not Ready (RNR)	23
4.4	LLC Parameter Format	24
4.5	Parameter Descriptions	24
4.5.1	Version Number, VERSION	25
4.5.2	Maximum Information Unit Extension, MIUX	25
4.5.3	Well-Known Service List, WKS	26
4.5.4	Link Timeout, LTO	26

4.5.5	Receive Window Size, RW	27
4.5.6	Service Name, SN.....	28
4.5.7	Option, OPT.....	28
5	LLC Procedures.....	30
5.1	MIU Requirements	30
5.2	Link Activation Procedure	30
5.2.1	Exchange of PAX PDU	30
5.2.2	LLCP Version Number Agreement Procedure.....	31
5.2.3	Link MIU Determination Procedure.....	31
5.3	Normal Operation Phase.....	32
5.4	Link Deactivation Procedure.....	32
5.4.1	Intentional Link Deactivation	32
5.5	Connectionless Transport Mode Procedures	32
5.5.1	Logical Data Link Parameters	32
5.5.2	Information Transfer.....	32
5.6	Connection-oriented Transport Mode Procedures.....	33
5.6.1	Data Link Connection State Variables.....	33
5.6.2	Data Link Connection Parameters	33
5.6.3	Connection Establishment	34
5.6.4	Information Transfer.....	35
5.6.5	Receiver Busy Condition.....	36
5.6.6	Connection Termination	37
5.7	Aggregation Procedure.....	37
5.8	Symmetry Procedure	37
6	MAC Mappings.....	39
6.1	Requirements.....	39
6.2	Definition of the ISO18092 MAC Mapping.....	39
6.2.1	Restrictions on the use of ISO 18092	39
6.2.2	Restrictions on LLC when using ISO 18092	39
6.2.3	MAC Link Activation.....	40
6.2.4	MAC Link Deactivation	41
6.2.5	Normal Operation.....	42
6.2.6	Error Handling.....	42

Figures

Figure 1: Relationship to OSI Reference Model	4
Figure 2: Logical Components	11
Figure 3: Format of the LLC PDU	14
Figure 4: Format of the SYMM PDU.....	17
Figure 5: Format of the PAX PDU.....	17
Figure 6: Format of the AGF PDU.....	18
Figure 7: Encapsulated LLC PDU format.....	18
Figure 8: Format of the UI PDU.....	18
Figure 9: Format of the CONNECT PDU	19

Figure 10: Format of the DISC PDU	19
Figure 11: Format of the CC PDU	20
Figure 12: Format of the DM PDU	20
Figure 13: Format of the FRMR PDU	21
Figure 14: Format of the I PDU	22
Figure 15: Format of the RR PDU	23
Figure 16: Format of the RNR PDU	23
Figure 17: LLC Parameter Format	24
Figure 18: Format of the VERSION parameter TLV	25
Figure 19: Format of the MIUX parameter TLV	26
Figure 20: Format of the WKS parameter TLV	26
Figure 21: Format of the LTO parameter TLV	27
Figure 22: Format of the RW parameter TLV	27
Figure 23: Format of the Service Name parameter TLV	28
Figure 24: Format of the Option parameter TLV	29
Figure 25: Format of the Attribute Request (ATR_REQ)	40
Figure 26: Format of the Attribute Response (ATR_RES)	41

Tables

Table 1: DSAP, SSAP Values	15
Table 2: PDU Type Values	15
Table 3: Disconnected Mode Reasons	20
Table 4: Frame Reject Information Fields	21
Table 5: Use of LLC Parameters	24
Table 6: Link Service Classes	29
Table 7: FRMR Error Conditions	36
Table 8: Revision History	43

1 Overview

This document describes the functions, features and protocol of the Logical Link Control (LLC) layer of the NFC Forum Protocol Suite. The LLC layer constitutes the upper half of the OSI data link layer (cf. figure 1) which is complemented by the lower half, the Medium Access and Control (MAC) layer. MAC layers are supported by the LLCP specification through a set of mappings. Each mapping specifies the binding requirements for LLCP to one externally defined MAC protocol.

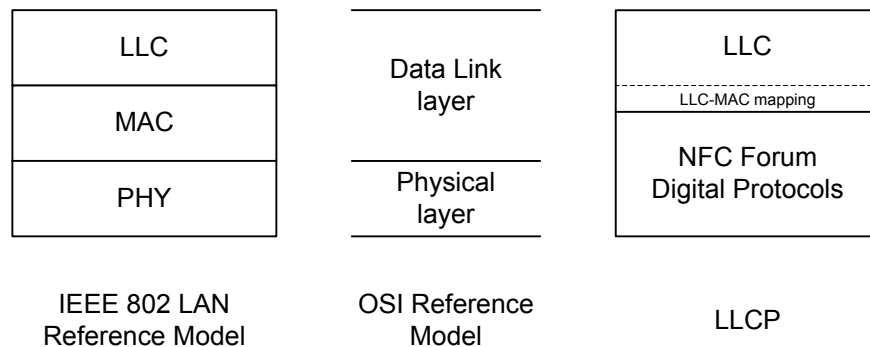


Figure 1: Relationship to OSI Reference Model

1.1 Features and Scope

The Logical Link Control Protocol (LLCP) specification provides the procedural means for the transfer of upper layer information units between two NFC Forum Devices. The set of procedures represents a uniform link abstraction to the user of the data link service.

The main features provided by the LLCP are:

- Link Activation, Supervision and Deactivation

The LLCP specifies how two NFC Forum Devices within communication range recognize compatible LLCP implementations, establish an LLCP Link, supervise the connection to the remote peer device, and deactivate the link if requested.

- Asynchronous Balanced Communication

Typical NFC MACs operate in Normal Response Mode where only a master, called the Initiator, is allowed to send data to and request data from the slave, called the Target. The LLCP enables Asynchronous Balanced Mode (ABM) between service endpoints in the two peer devices by use of a symmetry mechanism. Using ABM, service endpoints may initialize, supervise, recover from errors and send information at any time.

- Protocol Multiplexing

The LLCP is able to accommodate several instances of higher level protocols at the same time.

- Connectionless Transport

With minimum protocol overhead, connectionless transport provides a service user an unacknowledged data transmission facility that does not require preparative steps to actually send service data units. This transport mode can be used if upper protocol layers implement their own flow control and so need not rely on the link layer flow control mechanism. It can also be used by applications that operate in a command/response model wherein a command is always followed by a response returned before the next command is sent.

- Connection-oriented Transport

This transport mode provides a data transmission service with sequenced and guaranteed delivery of service data units. Traffic is controlled by a numbering scheme known as the sliding window protocol. Connection-oriented transport requires the preliminary setup of a data link connection and the assignment of resources for as long as the connection persists.

The following features are not defined or are not within the scope of this LLCP specification:

- Isochronous Data Transfer

The LLCP is not designed to transfer constant-rate data such as speech or video. No timing guarantees are given with respect to the transmission or reception of service data units and no time synchronization between sender and receiver is provided.

- Multicast / Broadcast

The LLCP does not support multicast or broadcast destination addressing. Data units are always destined for a single service access point.

- Secure Data Transfer

The LLCP does not provide secure data transfer between any two service access points. Secure communication may be provided by the lower or upper layer protocols.

- Service User Interface

The LLCP does not specify the interface between the LLC layer and the upper protocol layer. Implementers may choose any appropriate means of device software integration. An interoperable interface may be defined by other NFC Forum specifications.

- Implementation Architecture

The LLCP does not impose any internal functional partitioning nor does it mandate any internal interface definitions or component clustering.

1.2 Applicable Documents or References

1.2.1 Normative Authorities

[DIGIPROT]	NFC Forum. n.d. <i>NFC Digital Protocol Specification</i> . Working Draft. Forthcoming. Wakefield, Mass.: NFC Forum
[ISO18092]	International Organization for Standardization. 2004. <i>Near Field Communication – Interface and Protocol</i> . ISO/IEC 18092:2004. First Edition. Geneva: ISO
[RFC2119]	S. Bradner. 1997. <i>Key words for use in RFCs to Indicate Requirement Levels</i> . Internet Engineering Task Force RFC 2119. IETF

1.2.2 Normative References

[NFCREG] NFC Forum. n.d. *NFC Forum Assigned Values*. Forthcoming. Wakefield, Mass.: NFC Forum

1.2.3 Informative References

[BLACK] Black, Uyless. 1993. *Data Link Protocols*. Englewood Cliffs, N.J.: PTR Prentice-Hall

1.3 Administration

The NFC Forum Logical Link Control Protocol Specification is an open specification supported by the Near Field Communication Forum, Inc., located at:

401 Edgewater Place, Suite 600
Wakefield, MA, 01880

Tel.: +1 781-876-6216
Fax: +1 781-610-9864

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

The NFC Forum Devices Technical Working Group maintains this specification. Comments, errors, and other feedback can be submitted at <http://www.nfc-forum.org/apps/org/workgroup/chairs/download.php/6070>.

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 [RFC2119].

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 Logical Link Control Protocol 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, and revised on August 4, 2009.

1.7 Glossary

Connectionless Transport

An unacknowledged data transmission service with minimal protocol complexity.

Connection-oriented Transport

A data transmission service with sequenced and guaranteed delivery of service data units.

Data Link Connection

A unique combination of source and destination service access point address used for numbered information transfer.

ISO/IEC 18092

Near field communication protocol according to ISO/IEC 18092. See [DIGIPROT] and [ISO18092].

Link MIU

The maximum number of octets in the information field of any possible LLC protocol data unit.

Local LLC

The LLC component running on the local device.

Logical Data Link

A combination of source and destination service access point address used for unnumbered information transfer.

LLCP Link

The reliable communication channel between the local and the remote LLC that provides the transport for all data link connections and logical data links.

Maximum Information Unit (MIU)

The maximum length of the information field in a single LLC protocol data unit.

Protocol Data Unit (PDU)

The sequence of contiguous octets delivered as a unit to the adjacent lower layer or received as a unit from the adjacent lower layer.

Remote LLC

The LLC component running on the remote device.

Service

The capabilities and features provided to the adjacent upper layer.

Service Data Unit (SDU)

The sequence of contiguous octets received as a unit from the adjacent upper layer or delivered as a unit to the adjacent upper layer.

Service Discovery Protocol (SDP)

An application protocol to discover service access point users bound at the remote LLC.

Service User

The user of a service provided by the adjacent lower layer.

Type-Length-Value (TLV)

A coding method for parameters where the Type field specifies the parameter type, the Length field specifies the length of the parameter value and the Value field contains the actual parameter value octets

1.8 Acronyms and Abbreviations

ABM	Asynchronous Balanced Mode
AGF	Aggregated Frame
ATR_RES	Attribute Response
ATR_REQ	Attribute Request
CC	Connection Complete
CONNECT	Connect
DEP_RES	Data Exchange Protocol Response
DEP_REQ	Data Exchange Protocol Request
DID	Device Identifier

DISC	Disconnect
DM	Disconnected Mode
DSAP	Destination Service Access Point
FRMR	Frame Reject
I	Information
LLC	Logical Link Control
LLCP	Logical Link Control Protocol
MAC	Media Access Control
MIU	Maximum Information Unit
MIUX	Maximum Information Unit Extension
N(R)	Receive Sequence Number
N(S)	Send Sequence Number
NAD	Node Address
PAX	Parameter Exchange
PDU	Protocol Data Unit
PTYPE	Protocol Data Unit Type
RNR	Receive Not Ready
RR	Receive Ready
RW	Receive Window
RWT	Response Waiting Time
SAP	Service Access Point
SDP	Service Discovery Protocol
SDU	Service Data Unit
SSAP	Source Service Access Point
SYMM	Symmetry
TLV	Type-Length-Value
UI	Unnumbered Information
V(R)	Receive State Variable
V(RA)	Receive Acknowledgement State Variable
V(S)	Send State Variable
V(SA)	Send Acknowledgement State Variable
WTX	Waiting Time Extension

2 Definitions

2.1 Bit Significance in Diagrams and Tables

Unless otherwise specified, this document assumes the left-most bit in field definitions is the most significant bit and the right-most bit is least significant bit.

2.2 LLC Protocol Version

This specification describes Version 1.0 of the LLC Protocol.

3 Architecture

3.1 Components

LLCP can be split into the following logical components:

- The *MAC Mapping* integrates an existing RF protocol - such as ISO18092 - into the LLCP architecture.
- The *Link Management* component is responsible for serializing all connection-oriented and connectionless LLC PDU exchanges and for aggregation and disaggregation of small PDUs. This component also guarantees asynchronous balanced mode communication and provides link status supervision by performing the symmetry procedure.
- The *Connection-oriented Transport* component is responsible for maintaining all connection-oriented data exchanges including connection set-up and termination.
- The *Connectionless Transport* component is responsible for handling unacknowledged data exchanges.

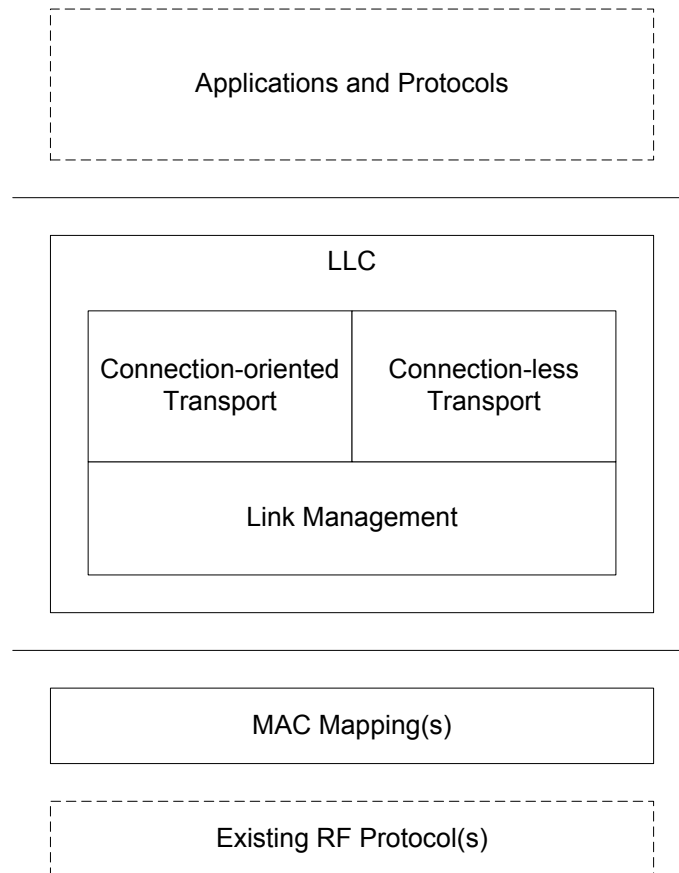


Figure 2: Logical Components

3.2 LLCP Data Transport Services

This section describes the different forms of data transport services that are provided to LLC service users. NFC Forum devices are classified according to their *link service class*—the combination of link transport services they offer to remote devices.

3.2.1 Link Service Class

This version of the LLCP specification defines two forms of data transport service: Connectionless transport and Connection-oriented Transport. Devices offering only the Connectionless transport service are classified as *link service class 1*. Devices offering only the Connection-oriented transport service are classified as *link service class 2*. Devices offering both Connectionless and Connection-oriented transport services are classified as *link service class 3*.

3.2.2 Connectionless Transport

Connectionless transport provides an unacknowledged data transmission service with minimum protocol complexity. This transport service is useful when higher layers provide any essential recovery and sequencing services so that replication of these services in the data link layer becomes needless. It may also prove useful in applications where guaranteed delivery of service data units is not essential. Due to the inherent data transfer reliability of the MAC layer, LLCP connectionless transport data units are guaranteed to arrive at the destination LLC but may not be delivered to the destination service access point.

Connectionless transport does not require any prior connection establishment or terminating procedure, thus allows spontaneous exchange of service data units without a time consuming preamble.

The source and destination of each connectionless service data unit are uniquely determined by the SSAP and DSAP field, respectively, in the data link PDU. Each combination of DSAP and SSAP forms a “logical data link” and this terminology is used in this specification to reference connectionless transport service.

3.2.3 Connection-oriented Transport

Connection-oriented transport provides a data transmission service with sequenced and guaranteed delivery of service data units. These types of data exchanges are described in this specification as “data link connections.”

Prior to the exchange of information-bearing PDUs, a data link connection is established between two service endpoints. The normal cycle of communication then consists of the transfer of PDUs containing information from one LLC to the other, acknowledged by PDUs in the opposite direction. An LLC acknowledges a PDU after it has been dispatched to the service user.

Information flow between the two peer LLCs is controlled by numbering the PDUs and managing a sliding window of sent, but unacknowledged, PDUs. A data link connection persists with its associated resources until explicit closure by either one of the two communicating LLCs or as a consequence of an unexpected link disruption.

Each data link connection is uniquely identified by the pair of DSAP and SSAP included in the header of each LLC PDU. When opening a connection, the DSAP identifies the remote service endpoint for the intended data exchange and the SSAP identifies the local service endpoint. Once the connection is established, each LLC sends PDUs for that data link connection by alternately exchanging the DSAP and SSAP values such that each sent PDU identifies the source LLC's SAP in the SSAP field and the destination LLC's SAP in the DSAP field. During the lifetime of a data link connection, both LLCs maintain state information and manage receive buffers independently of any other data link connection.

The number of receive buffers at both endpoints of a data link connection is implementation specific, but the PDU numbering scheme in this version of the LLC specification does not support more than 15 unacknowledged numbered information units. Lower numbers can be announced during connection setup.

4 LLC Elements

This section defines in detail the LLC PDU structure, the method for representing data link layer service access point addresses, the available LLC PDU types and how they are indicated, and the use of the control and information fields for specific PDUs.

4.1 LLC PDU Format

All LLC PDUs SHALL conform to the format shown in figure 3:

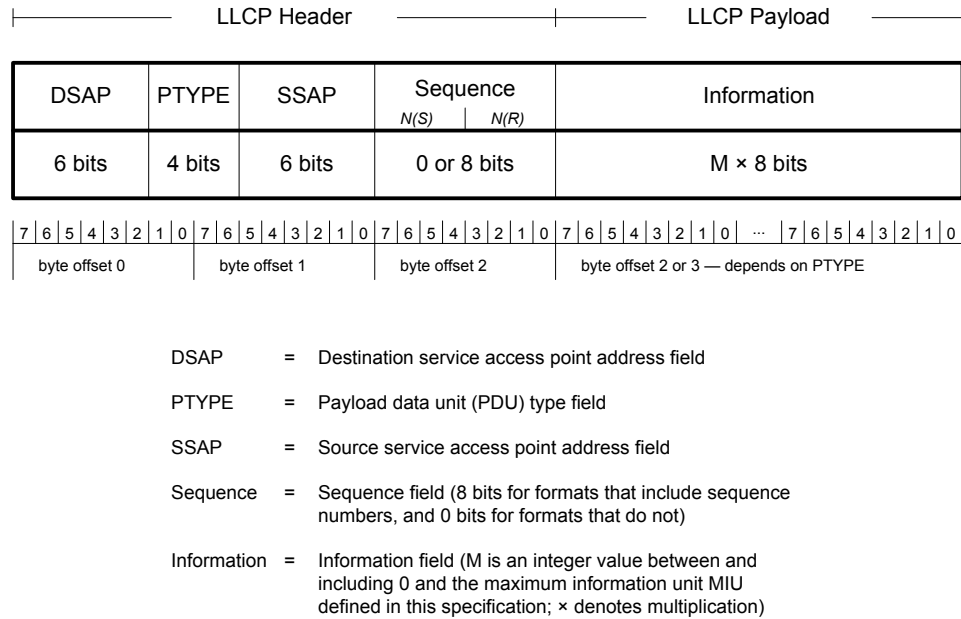


Figure 3: Format of the LLC PDU

4.2 Elements of the LLC PDU

4.2.1 Address Fields

Each LLC PDU SHALL contain two address fields – the Destination Service Access Point (DSAP) address field and the Source Service Access Point (SSAP) address field, in the bit positions shown in figure 3. The DSAP address field SHALL identify the service access point for which the LLC information field is intended. The SSAP address field SHALL identify the service access point from which the LLC information field originated.

4.2.1.1 Address Usage

Any address SHALL be usable as both an SSAP and a DSAP address. Table 1 lists the different values that can be taken by the DSAP and SSAP fields and a brief description for the services that are identified by these values.

Table 1: DSAP, SSAP Values

DSAP/SSAP	Description
00h – 0Fh	Identifies the Well-Known Service Access Points
10h – 1Fh	Identifies Services in the local service environment and are advertised by local SDP
20h – 3Fh	Identifies Services in the local service environment and are NOT advertised by local SDP

The “0” address, i.e. 00h, in the DSAP or SSAP address field SHALL designate the LLC Link Management component, and SHALL NOT be used to identify any specific service access point.

The “1” address, i.e. 01h, SHALL designate the well-known service access point for the Service Discovery Protocol (SDP).

Address values between 2 and 15 (02h – 0Fh) inclusively, SHALL be reserved for well-known service access points defined in the NFC Forum Assigned Values Registry [NFCREG].

Address values between 16 and 31 (10h – 1Fh) inclusively, SHALL be assigned by the local LLC to services registered by the local service environment. These registrations SHALL be made available by the local Service Discovery Protocol (SDP) instance for discovery and use by a remote LLC.

Address values between 32 and 63 (20h – 3Fh) inclusively, SHALL be assigned by the local LLC as the result of an upper layer service request, and SHALL NOT be available for discovery using the Service Discovery Protocol (SDP).

4.2.2 PDU Type

Each LLC PDU SHALL contain a 4-bit protocol data unit type field—the PTYPE field—in the bit positions shown in figure 3. The PTYPE field SHALL identify the syntax and semantic of the remaining fields of the LLC PDU, and SHALL take one of the values specified in Table 2. Values marked as reserved in Table 2 SHALL NOT be generated by the sender and SHALL be ignored by the receiver.

In Table 2, the PTYPE values for each PDU Type are defined as are the link service classes for which each PDU Type is applicable.

The PTYPE value 1111b SHALL be reserved for future extensions of the LLCP header format and SHALL then be used in conjunction with the DSAP and SSAP fields set to “0”. The extension header format can be recognized by the leading two octets “03h C0h”, in that order. Implementations conforming to this version SHALL treat this value as a reserved PTYPE.

Table 2: PDU Type Values

PDU Type	PTYPE	Link Service Class
SYMM	0000	1, 2, 3
PAX	0001	1, 2, 3
AGF	0010	1, 2, 3
UI	0011	1, 3

PDU Type	PTYPE	Link Service Class
CONNECT	0100	2, 3
DISC	0101	1, 2, 3
CC	0110	2, 3
DM	0111	1, 2, 3
FRMR	1000	2, 3
reserved	1001	
reserved	1010	
reserved	1011	
I	1100	2, 3
RR	1101	2, 3
RNR	1110	2, 3
reserved	1111	

4.2.3 Sequence Field

The sequence field SHALL be present in PDUs containing sequence numbers, in the bit positions shown in figure 3. It SHALL NOT be present in other PDUs. The sequence field is always divided into a 4 bit sub-field containing send sequence number N(S) encoded in the most significant bits, and a 4-bit sub-field containing receive sequence number N(R) encoded in the least significant bits. Section 4.3 specifies, for each PDU, whether the sequence field is present and how the sub-fields are utilized.

In this version of the LLCP specification, PDUs containing sequence numbers are the I, RR, and RNR PDU.

4.2.4 Information Field

The information field SHALL consist of an integral number of up to MIU octets. The information field MAY be empty. Section 4.3 specifies, for each PDU, whether the information field is present and what it contains.

4.3 PDU Descriptions

This subsection defines the structure of all valid LLC PDUs.

4.3.1 Symmetry (SYMM)

The SYMM PDU is an unnumbered PDU sent by an LLC whenever no other PDUs are available for sending, to ensure symmetry.

The format of the SYMM PDU SHALL be as shown in figure 4.

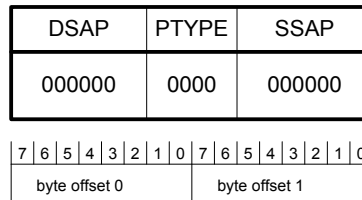


Figure 4: Format of the SYMM PDU

The sequence field SHALL NOT be present in a SYMM PDU.

An information field SHALL NOT be present in a SYMM PDU.

4.3.2 Parameter Exchange (PAX)

The PAX PDU SHALL be used to exchange parameters concerning the LLC Link configuration.

The format of the PAX PDU SHALL be as shown in figure 5 (where n is the length of the PDU).

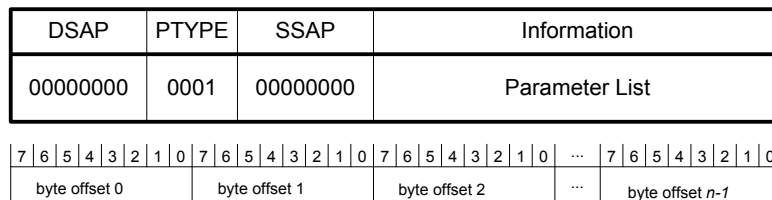


Figure 5: Format of the PAX PDU

The sequence field SHALL NOT be present in a PAX PDU.

PAX SHALL NOT use any DSAP or SSAP value other than 0.

The information field of the PAX PDU SHALL be encoded as a contiguous sequence of LLC parameter TLV elements, with the generic encoding format defined in section 4.4. The LLC parameters that are allowed in the information field of a PAX PDU are defined in section 4.5. The receiver of a PAX PDU SHALL ignore any TLV elements that are not defined in section 4.5.

4.3.3 Aggregated Frame (AGF)

The AGF is an unnumbered PDU. The AGF PDU MAY be used by the LLC Link Management component to aggregate and transfer multiple LLC PDUs to the remote LLC Link Management component in a single transmission.

Upon reception of an AGF PDU, the LLC Link Management component SHALL dispatch the aggregated LLC PDUs according to each PDU's destination and source service access point address. Each PDU SHALL be dispatched as if transmitted separately and in the order it appears in the AGF PDU.

The format of the AGF PDU SHALL be as shown in figure 6 (where n is the length of the PDU).

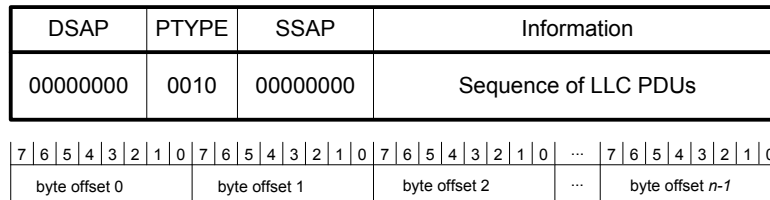


Figure 6: Format of the AGF PDU

The sequence field SHALL NOT be present in an AGF PDU.

The information field of the AGF PDU SHALL be encoded as a contiguous sequence of LLC PDUs. Each PDU SHALL be encapsulated by preceding it with a 16-bit length field, encoded most significant byte first, as shown in figure 7. The length field SHALL specify the number of octets in the following LLC PDU.

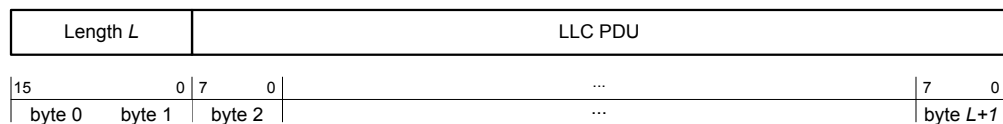


Figure 7: Encapsulated LLC PDU format

The AGF PDU SHALL contain at least two encapsulated PDUs. SYMM and AGF PDUs SHALL NOT be aggregated. No other restrictions exist on the type of PDU, or the use of service access point addresses, or the values of sequence numbers.

The maximum number of octets in the information field of the AGF PDU SHALL be determined by the Link MIU established with the LLC link activation procedure described in section 5.2.

4.3.4 Unnumbered Information (UI)

The UI PDU is used to transfer service data units to the peer LLC without prior establishment of a data link connection.

The format of the UI PDU SHALL be as shown in figure 8.

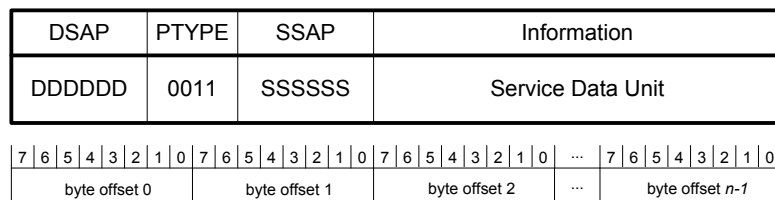


Figure 8: Format of the UI PDU

The sequence field SHALL NOT be present in a UI PDU.

The information field of a UI PDU SHALL contain a single service data unit. The maximum number of octets in the information field SHALL be determined by the Link MIU established with the LLC link activation procedure described in section 5.2.

The information field of a UI PDU MAY be empty.

4.3.5 Connect (CONNECT)

The CONNECT PDU is an unnumbered PDU which is used to request a data link connection between a source and a destination service access point.

The CONNECT PDU SHALL have the format shown in figure 9.

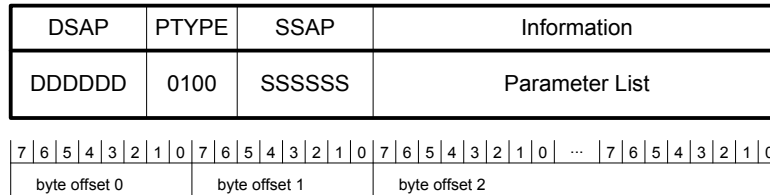


Figure 9: Format of the CONNECT PDU

The sequence field SHALL NOT be present in a CONNECT PDU.

The information field of the CONNECT PDU MAY contain connection specific parameters. These parameters SHALL be encoded as a contiguous sequence of TLV elements and each TLV element SHALL be encoded as defined in section 4.4. The LLC parameters that are allowed in the information field of a CONNECT PDU are defined in section 4.5. The receiver of a CONNECT PDU SHALL ignore any TLV elements that are not defined in section 4.5.

4.3.6 Disconnect (DISC)

The DISC PDU is an unnumbered PDU which is used to terminate a data link connection or is used to deactivate the LLCP Link.

The DISC PDU SHALL have the format shown in figure 10.

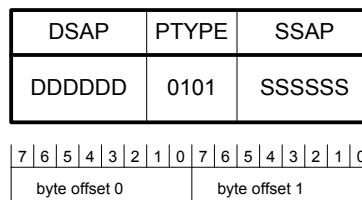


Figure 10: Format of the DISC PDU

The sequence field SHALL NOT be present in a DISC PDU.

An information field SHALL NOT be present in a DISC PDU.

4.3.7 Connection Complete (CC)

The CC PDU is an unnumbered PDU which is used by an LLC to acknowledge the receipt and acceptance of the CONNECT.

The CC PDU SHALL have the format shown in figure 11.

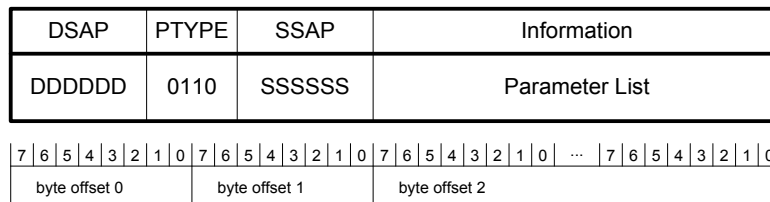


Figure 11: Format of the CC PDU

The sequence field SHALL NOT be present in a CC PDU.

The information field of the CC PDU MAY contain connection specific parameters. These parameters SHALL be encoded as a contiguous sequence of TLV elements and each TLV element SHALL be encoded as defined in section 4.4. The parameters that are allowed in the information field of a CC PDU are defined in section 4.5. The receiver of a CC PDU SHALL ignore any TLV elements that are not defined in section 4.5.

4.3.8 Disconnected Mode (DM)

The DM PDU is an unnumbered PDU which is used to report status indicating that the LLC is logically disconnected from the data link connection identified by the DSAP and SSAP address pair.

The DM PDU SHALL have the format shown in figure 12.

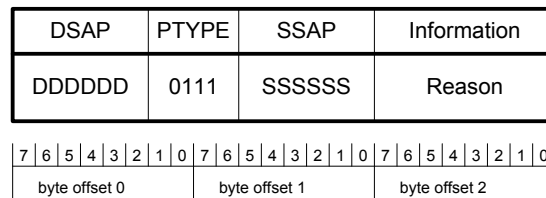


Figure 12: Format of the DM PDU

The sequence field SHALL NOT be present in a DM PDU.

The information field of the DM PDU SHALL contain a single octet, the reason field, that gives further information about the cause of the DM response. The reason field SHALL be one of the values in table 3. In table 3, the term *target service access point* refers to the value in the SSAP field of the DM PDU (which is the same as the destination access point specified in the PDU which caused return of the DM PDU).

Table 3: Disconnected Mode Reasons

00h	SHALL indicate that the LLC has received a DISC PDU and is now logically disconnected from the data link connection.
01h	SHALL indicate that the LLC has received a connection-oriented PDU but the target service access point has no active connection.

02h	SHALL indicate that the remote LLC has received a CONNECT PDU and there is no service bound to the specified target service access point.
03h	SHALL indicate that the remote LLC has processed a CONNECT PDU and the request to connect was rejected by the service layer.
10h	SHALL indicate that the LLC will permanently not accept any CONNECT PDUs with the same target service access point address.
11h	SHALL indicate that the LLC will permanently not accept any CONNECT PDUs with any target service access point address.
20h	SHALL indicate that the LLC will temporarily not accept any CONNECT PDUs with the specified target service access point address.
21h	SHALL indicate that the LLC will temporarily not accept any CONNECT PDUs with any target service access point address.
Any other value	SHALL NOT be used by an LLC sending a DM PDU, and SHALL be interpreted as 00h by an LLC receiving a DM PDU.

4.3.9 Frame Reject (FRMR)

The FRMR PDU is an unnumbered PDU which is used to report the receipt of a malformed or inappropriate PDU on a data link connection. The specific reasons for sending the FRMR PDU are described in section 5.6.4.5.

The FRMR PDU SHALL have the format shown in figure 13.

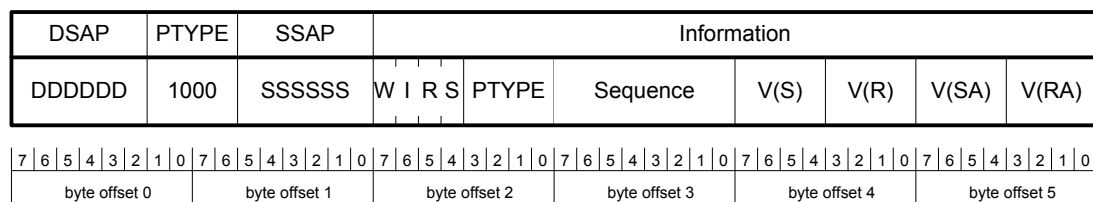


Figure 13: Format of the FRMR PDU

The sequence field SHALL NOT be present in a FRMR PDU.

An information field SHALL be returned with the FRMR PDU to provide the reason for the PDU rejection, and SHALL contain the fields shown in figure 13 with the following meaning:

Table 4: Frame Reject Information Fields

W Flag	Well-formedness Error—set to “1” SHALL indicate that the rejected PDU was invalid or not well formed. Please refer to clause 5.5.4.5 for further information.
I Flag	Information Field Error—set to “1” SHALL indicate that the rejected PDU contained an incorrect or unexpected information field. Please refer to clause 5.5.4.5 for further information.
R Flag	Receive Sequence Error—set to “1” SHALL indicate that the rejected PDU contained an invalid N(R) in the sequence field.

S Flag	Send Sequence Error—set to “1” SHALL indicate that the rejected PDU contained an invalid N(S) in the sequence field.
PTYPE field	SHALL indicate the type of the rejected PDU, i.e. it SHALL have the same value as the PTYPE field of the rejected PDU.
Sequence field	SHALL have the same value as the sequence field of the rejected PDU if the format of the rejected PDU is defined to contain sequence numbers. If the rejected PDU does not contain sequence numbers, i.e. the sequence field of the rejected PDU has a length of 0 bits; it SHALL be set to all bits zero.
V(S) field	SHALL contain the current value of the send state variable for this data link connection at the rejecting LLC.
V(R) field	SHALL contain the current value of the receive state variable value for this data link connection at the rejecting LLC.
V(SA) field	SHALL contain the current value of the acknowledgement send state variable for this data link connection at the rejecting LLC.
V(RA) field	SHALL contain the current value of the receive acknowledgement state variable value for this data link connection at the rejecting LLC.

4.3.10 Information (I)

The I PDU is a numbered PDU which is used to transfer service data units across a data link connection. The format of the I PDU SHALL be as shown in figure 14, where n is the length of the I PDU.

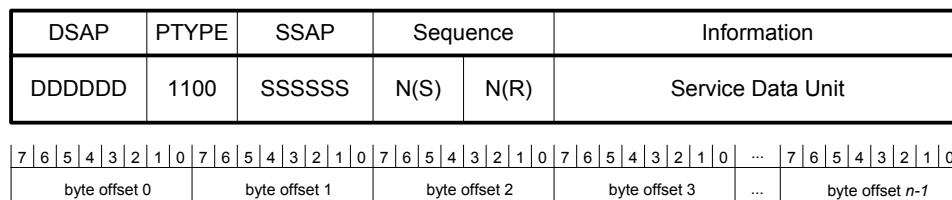


Figure 14: Format of the I PDU

The I PDU sequence field SHALL contain two sequence numbers: The send sequence number N(S) and the receive sequence number N(R). The send sequence number N(S) SHALL indicate the sequence number associated with this I PDU. The receive sequence number N(R) value SHALL indicate that I PDUs numbered up through N(R) - 1 have been received correctly by the sender of this I PDU and successfully passed to the senders SAP identified in the SSAP field. These I PDUs SHALL be considered as acknowledged.

The information field of an I PDU SHALL contain a single service data unit. The maximum number of octets in the information field SHALL be determined by the MIU for the data link connection as described in section 5.6.2.1.

The information field of an I PDU MAY be empty.

4.3.11 Receive Ready (RR)

The RR PDU is a numbered PDU which is used by an LLC to acknowledge one or more received I PDUs and indicate that the LLC is able to receive subsequent I PDUs.

The RR PDU SHALL have the format shown in figure 15.

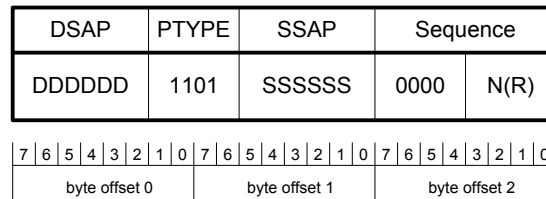


Figure 15: Format of the RR PDU

The RR PDU sequence field SHALL contain the receive sequence number N(R). The receive sequence number N(R) SHALL indicate that I PDUs numbered up through N(R) - 1 have been received correctly by the sender of this I PDU and successfully passed to the senders SAP identified in the SSAP field. These I PDUs SHALL be considered as acknowledged. The higher nibble of the sequence field SHALL be set to zero by the sender and SHALL be ignored by the receiver.

An information field SHALL NOT be present in an RR PDU.

4.3.12 Receive Not Ready (RNR)

The RNR PDU is a numbered PDU which is used by an LLC to indicate a temporary inability to process subsequent I PDUs.

The RNR PDU SHALL have the format shown in figure 16.

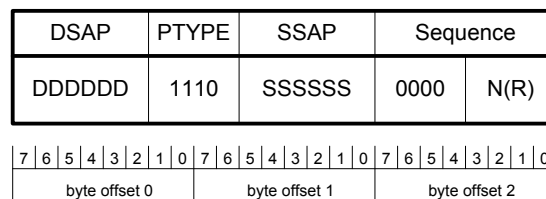


Figure 16: Format of the RNR PDU

The RR PDU sequence field SHALL contain the receive sequence number N(R). The receive sequence number N(R) SHALL indicate that I PDUs numbered up through N(R) - 1 have been received correctly by the sender of this I PDU and successfully passed to the senders SAP identified in the SSAP field. These I PDUs SHALL be considered as acknowledged. The higher nibble of the sequence field SHALL be set to zero by the sender and SHALL be ignored by the receiver.

An information field SHALL NOT be present in an RNR PDU.

4.4 LLC Parameter Format

All LLC Parameters SHALL conform to the format shown in figure 17.

Type	Length	Value			
XXh	<i>n</i>	1 st value octet	...	<i>n</i> th value octet	
7 byte offset 0	0 byte offset 1	7 byte offset 2	...	7 byte offset <i>n</i> +1	0

Figure 17: LLC Parameter Format

4.5 Parameter Descriptions

This subsection defines the syntax and semantic of all LLC parameters that may appear in the information field of LLC PDUs or in MAC specific link activation frames.

A parameter exchanged using LLC PDUs SHALL appear in the information field of only the LLC PDUs listed in table 5.

Any parameter that can be exchanged in the information field of a PAX PDU MAY also be included in a MAC link activation message as defined by the specific MAC mapping.

Any parameters included in MAC link activation frames SHALL NOT be repeated in the PAX PDU used in the LLC link activation procedure (cf. section 5.2).

A specific MAC mapping MAY provide an alternative format for encoding LLC parameters in link activation frames.

A specific MAC mapping MAY prohibit the use of PAX PDU exchange during LLC link activation if it provides for the exchange of all LLC parameters allowed in a PAX PDU (shown in table 5) as part of the MAC link activation.

Table 5: Use of LLC Parameters

Parameter	Included	In PDU Type
Version Number, VERSION	MUST	PAX
Maximum Information Unit Extension, MIUX	MAY	PAX, CONNECT, CC
Well-Known Service List, WKS	SHOULD	PAX
Link Timeout, LTO	MAY	PAX
Receive Window Size, RW	MAY	CONNECT, CC
Service Name, SN	MAY	CONNECT
Option, OPT	MAY	PAX

In the event that a TLV is received which does not comply to the normative requirements of this section it SHALL be ignored.

4.5.1 Version Number, VERSION

The version number (VERSION) parameter is a configuration parameter that **MUST** be transmitted during the link activation. It **SHALL** denote the major and minor release levels of the LLC specification implemented by the sending LLC.

The VERSION parameter **SHALL** be encoded as an 8-bit structure consisting of two 4-bit unsigned integer values representing the major and minor release levels of this specification. The most-significant 4 bits **SHALL** denote the major release level. The least-significant 4 bits **SHALL** denote the minor release level of this specification.

When the VERSION parameter is encoded as a TLV, the TLV Type field **SHALL** be 01h and the TLV Length field **SHALL** be 01h.

Type	Length	Value	
01h	01h	Major	Minor

7	0	7	0	7	4	3	0
byte 0		byte 1		byte 2			

Figure 18: Format of the VERSION parameter TLV

4.5.2 Maximum Information Unit Extension, MIUX

The maximum information unit (MIU) is the maximum number of octets in the information field of an LLC PDU that the local LLC is able to receive. The default MIU is 128.

The maximum information unit extension (MIUX) parameter **MAY** be used by an LLC to announce the ability to receive LLC PDUs with information fields larger than the default.

The MIUX parameter **MAY** be transmitted during link activation to announce the local LLC's larger Link MIU.

The MIUX parameter **MAY** be transmitted in the information field of a CONNECT or CC PDU to announce the local LLC's larger MIU for that data link connection endpoint.

The MIUX parameter specifies the number of octets by which an information field **MAY** exceed the default maximum size. It **SHALL** be encoded as an 11-bit unsigned integer value according to the following formula:

$$MIUX = MIU - 128$$

When the MIUX parameter is encoded as a TLV, the TLV Type field **SHALL** be 02h and the TLV Length field **SHALL** be 02h. The MIUX parameter **SHALL** be encoded into the least-significant 11 bits of the TLV Value field. The unused bits in the TLV Value field **SHALL** be set to zero by the sender and **SHALL** be ignored by the receiver.

Type	Length	Value	
02h	02h	00000b	MIUX

7	0	7	0	15	11	10	0
byte 0		byte 1		byte 2		byte 3	

Figure 19: Format of the MIUX parameter TLV

The receiver of an MIUX parameter SHALL calculate the MIU value as follows:

$$MIU = 128 + MIUX$$

If no MIUX parameter is transmitted, the default MIU value of 128 SHALL be used.

4.5.3 Well-Known Service List, WKS

The well-known service (WKS) parameter is a configuration parameter that SHALL denote the binding of service listeners to well-known service access point addresses and therefore the willingness of the sender of the WKS parameter to accept PDUs on those SAPs.

The WKS parameter SHALL be encoded as a 16-bit field. The most-significant bit of the 16-bit field value SHALL signify SAP address 0Fh and the least-significant bit SHALL signify SAP address 00h. The other bits SHALL signify SAP addresses corresponding to their respective bit positions. A bit set to “1” SHALL indicate that a service listener is bound to the corresponding well-known service access point. A bit set to “0” SHALL indicate that no service listener is bound to the corresponding well-known service access point. As the LLC Link Management Service is bound to SAP 00h, the least significant bit, representing SAP 00h, SHALL always be set to “1” by the sender and SHALL be ignored by the receiver.

When the WKS parameter is encoded as a TLV, the TLV Type field SHALL be 03h, the TLV Length field SHALL be 02h, and the WKS parameter value SHALL be encoded into the TLV Value field with the most-significant byte first.

Type	Length	Value	
03h	02h	XXXXXXXXXXXXXX1b	

7	0	7	0	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
byte 0		byte 1		byte 2		byte 3													

Figure 20: Format of the WKS parameter TLV

4.5.4 Link Timeout, LTO

The link timeout (LTO) parameter is a configuration parameter that specifies the local link timeout interval guarantee.

The LTO parameter SHALL specify the maximum time interval between the last received bit of an LLC PDU transmission from the remote to the local LLC and the first bit of the subsequent

LLC PDU transmission from the local to the remote LLC. It is an implementation concern to take into account any internally required processing time and propagation delays.

The LTO parameter value SHALL be an 8-bit unsigned integer that specifies the link timeout value in multiples of 10 milliseconds.

When the LTO parameter is encoded as a TLV, the TLV Type field SHALL be 04h and the TLV Length field SHALL be 01h.

If no LTO parameter is transmitted or if the LTO parameter value is zero, the default link timeout value of 100 milliseconds SHALL be used.

Type	Length	Value
04h	01h	LTO

7	0	7	0	7	0
byte 0		byte 1		byte 2	

Figure 21: Format of the LTO parameter TLV

4.5.5 Receive Window Size, RW

The receive window size (RW) is a data link connection parameter that MAY be transmitted with a CONNECT or a CC PDU and applies to the sender of the CONNECT or CC PDU.

The RW parameter SHALL be encoded as a 4-bit unsigned integer value indicating the receive window size. The receive window size SHALL be in the inclusive range of values between 0 and 15.

Note: A receive window size of zero indicates that the local LLC will not accept I PDUs on that data link connection. A receive window size of one indicates that the local LLC will acknowledge every I PDU before accepting additional I PDUs.

To encode the RW parameter as a TLV, the TLV Type field SHALL be 05h and the TLV Length field SHALL be 01h. The RW parameter SHALL be encoded into the least-significant 4 bits of the TLV Value field. The unused bits in the TLV Value field SHALL be set to zero by the sender and SHALL be ignored by the receiver.

If not transmitted, the default value of RW SHALL be 1.

Type	Length	Value
05h	01h	0000 RW

7	0	7	0	7	6	5	4	3	0
byte 0		byte 1		byte 2					

Figure 22: Format of the RW parameter TLV

4.5.6 Service Name, SN

The service name (SN) parameter MAY be transmitted with a CONNECT PDU to the well-known destination service access point address 01h and SHALL then indicate that the sending LLC intends to establish a data link connection with the named service registered in the remote service environment.

To encode the service name parameter as a TLV, the TLV Type field SHALL be set to 06h and the TLV Length field SHALL be set to the number of octets within the TLV Value field.

If the service name parameter is transmitted with a CONNECT PDU to a destination service access point other than 01h, it SHALL be ignored.

Type	Length	Value
06h	n	Service Name URI

7	0	7	0	7	6	5	4	3	2	1	0	...	7	6	5	4	3	2	1	0
byte 0				byte 1				byte offset 2				...	byte offset $n+1$							

Figure 23: Format of the Service Name parameter TLV

The format of the value field SHALL be a URI reference and SHALL represent either an NFC Forum well-known service or an external service name.

NFC Forum well-known service names SHALL take the form:

"urn:nfc:sn:<servicename>"

External service names MAY be formatted as uniform resource names (URN) with the reserved prefix "urn:nfc:xsn:" if the following requirements are met:

The service name SHALL be prefixed with the fully qualified domain name of the service owner followed by a colon (":") and followed by a service name of the service owner's choosing:

urn:nfc:xsn:<domain>:<servicename>

Examples:

"urn:nfc:xsn:somedomain.com:atestservice"

"urn:nfc:xsn:somedomain.com:md5:01234567890123456789012345678901"

4.5.7 Option, OPT

The Option (OPT) parameter is a configuration parameter that MAY be transmitted during the LLCP Link activation.

The Option parameter communicates the link service class and the set of options supported by the sending LLC.

To encode the Option parameter as a TLV, the TLV Type field SHALL be 07h and the TLV Length field SHALL be 01h.

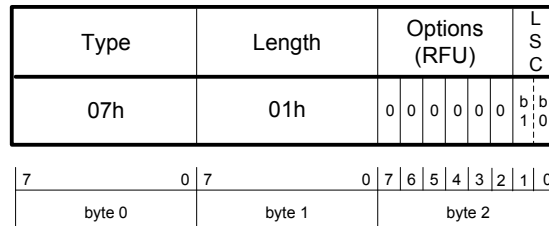


Figure 24: Format of the Option parameter TLV

The Value field contains a single 8-bit byte representing a set of flags which indicate the link service class of the sending LLC and the support of optional features implemented by the sending LLC.

Link Service Class

Within the link service class subfield (LSC), the two bits, “*b0*” and “*b1*”, indicate support for the connectionless and connection-oriented link transports, respectively:

- Bit *LSC b0* indicates support in the sending LLC for Connectionless transport mode, as described in section 3.2.2 of this specification. The sending LLC SHALL set bit *LSC b0* to 1 when Connectionless transport is supported. The sending LLC SHALL set bit *LSC b0* to 0 to when Connectionless transport is not supported.
- Bit *LSC b1* indicates support in the sending LLC for Connection-oriented transport mode, as described in section 3.2.3 of this specification. The sending LLC SHALL set bit *LSC b1* to 1 when Connection-oriented transport is supported. The sending LLC SHALL set bit *LSC b1* to 0 when Connection-oriented transport is not supported.

The LSC bits form a 2-bit unsigned value that represents the link service class of the sending device as shown in Table 6.

When the Option parameter is not received during the LLCP Link activation, or if the link service class is specified as 0 (unknown), the link service class supported by the remote LLC remains unknown to the local LLC. In this case the local LLC MAY behave as though link service class 3 was specified, thus allowing the local LLC to attempt communication with the remote LLC using either transport service.

Table 6: Link Service Classes

Link Service Class	LSC <i>b1</i>	LSC <i>b0</i>	Definition
unknown	0	0	Link service class is unknown at time of link activation
Class 1	0	1	Connectionless link service only
Class 2	1	0	Connection-oriented link service only
Class 3	1	1	Both connectionless and connection-oriented link service

Other Options

All other bits are reserved for future use (RFU). An LLC MUST set the RFU bits to 0 when sending the Option parameter. An LLC receiving an Option parameter MUST ignore the values of the RFU bits.

5 LLC Procedures

5.1 MIU Requirements

The requirements of this section SHALL NOT apply to I PDUs.

The MIU value used for all LLC PDUs with information fields SHALL be the Link MIU determined according to section 5.2.3.

An LLC SHALL NOT send any LLC PDU with an information field that is larger than the Link MIU determined for the remote LLC.

An LLC MAY discard any received LLC PDU with an information field that is larger than the local LLCs Link MIU value.

5.2 Link Activation Procedure

The LLC link activation procedure SHALL be started when the local MAC component notifies the local LLC component that a peer device capable of executing LLCP enters communication range and the MAC link activation procedure has been successfully completed.

If the specific MAC mapping specifies the exchange of LLC parameters during the MAC link activation procedure, the parameters that have been received with the MAC link activation procedure SHALL be available to the local LLC.

If the specific MAC mapping does not prohibit the use of PAX PDU in LLC link activation, following successful completion of the MAC link activation, the PAX PDUs are exchanged according to section 5.2.1.

If the specific MAC mapping prohibits the use of PAX PDU in LLC link activation, the local LLC SHALL perform the version number agreement procedure defined in section 5.2.2:

- If version number agreement has been achieved, the local LLC SHALL perform the Link MIU agreement procedure defined in section 5.2.3 and SHALL then enter normal operation.
- If version number agreement cannot be achieved, the LLC SHALL abandon the link activation and SHOULD notify the MAC component that link activation failed.

5.2.1 Exchange of PAX PDU

The MAC link activation event will indicate whether the local LLC is operating in the Initiator or Target role.

Operating in the Initiator role:

The local LLC SHALL send a PAX PDU to the remote LLC that includes all required LLC parameters not exchanged during the MAC link activation. The local LLC SHALL then await receipt of a PAX PDU from the remote LLC. Upon receipt of the PAX PDU, the local LLC SHALL perform the version number agreement procedure defined in section 5.2.2.

- If version number agreement has been achieved, the local LLC SHALL perform the Link MIU agreement procedure defined in section 5.2.3 and SHALL then enter normal operation.
- If version number agreement cannot be achieved, the LLC SHALL abandon the link activation and SHOULD notify the MAC component that link activation failed.

Operating in the Target role:

The local LLC SHALL await receipt of a PAX PDU from the remote LLC. Upon receipt of the PAX PDU, the local LLC SHALL perform the version number agreement procedure defined in section 5.2.2.

- If the version number agreement has been achieved, the local LLC SHALL perform the Link MIU agreement procedure defined in section 5.2.3. The local LLC SHALL then return a PAX PDU to the remote LLC that includes all required LLC parameters that were not exchanged during the MAC link activation. The local LLC SHALL then enter normal operation.
- If version number agreement cannot be achieved, the LLC SHALL abandon the link activation and SHOULD notify the MAC component that link activation failed.

5.2.2 LLCP Version Number Agreement Procedure

The local LLC SHALL perform version number agreement to determine if the version of LLCP being used by the local LLC is compatible with the version of LLCP being used by the remote LLC.

The version of LLCP SHALL be described using a two-part version number comprised of a *major release value* and a *minor release value*. Version numbers SHALL be exchanged during LLC link activation using the VERSION parameter defined in section 4.5.1.

To determine version agreement, the local LLC SHALL compare the version information sent to the remote LLC with the version information it received from the remote LLC.

- In the case where the major release values are identical and the minor release values are identical then version agreement SHALL be achieved.
- In the case where the major release values are identical but the minor release values are not identical then version agreement SHALL be achieved. The agreed minor release value SHALL then be the lower of the two exchanged minor release values.
- In the case where the major release values are not identical then the LLC with the higher major release number SHALL decide if version agreement is possible. If version agreement is possible, the agreed LLCP version SHALL be the lower of the two version values exchanged (e.g. LLCs with versions 2.3 and 1.7 agree on version 1.7).

If version agreement is achieved, the local LLC SHALL behave according to the requirements of the agreed LLCP version.

5.2.3 Link MIU Determination Procedure

The local and remote LLCs each establish and maintain distinct Link MIU values.

The Link MIU prior to entering normal operation SHALL be the default MIU of 128 octets.

If the local LLC receives an MIUX parameter during link activation, the remote Link MIU SHALL be determined as specified in section 4.5.2.

If the local LLC does not receive an MIUX parameter during link activation, the remote Link MIU SHALL be determined to be the default MIU of 128 octets.

An LLC MAY send an MIUX parameter during link activation if it can accept LLC PDUs with information fields larger than the default MIU.

If an LLC does not send an MIUX parameter during link activation, the local Link MIU SHALL be the default value of 128 octets.

5.3 Normal Operation Phase

In the normal operation phase, the LLC MAY send or receive information using connectionless and connection-oriented transport mode procedures as defined in sections 5.5 and 5.6, MAY aggregate or disaggregate PDUs using the aggregation procedure defined in section 5.7, SHALL execute the symmetry procedure defined in section 5.8, and SHALL be able to perform the deactivation procedure defined in section 5.4 on request of a device management function.

When entering the normal operation phase in the Initiator role, the LLC SHALL initialize the symmetry procedure by assuming previous reception of an LLC PDU from the remote LLC. When entering the normal operation phase in the Target role, the LLC SHALL initialize the symmetry procedure by assuming previous transmission of an LLC PDU to the remote LLC.

5.4 Link Deactivation Procedure

The LLC link deactivation procedure SHALL be used to terminate normal operation between the local and the remote LLC. Either LLC MAY initiate link deactivation. The local LLC MUST execute link deactivation upon request from the remote LLC (cf. section 5.4.1) or as the result of a link timeout (cf. section 5.8).

When executing the link deactivation procedure, the local LLC SHOULD pass disconnect indications to the service layer and SHALL then execute the MAC link deactivation.

5.4.1 Intentional Link Deactivation

The LLC deactivation procedure MAY be initiated by sending a DISC PDU, with the DSAP and SSAP fields set to address “00h”, thus designating the LLC Link Management component.

After sending the DISC PDU, the LLC that initiated the link deactivation SHALL deactivate the local link.

When receiving a DISC PDU with the DSAP and SSAP fields set to address “00h”, the receiving LLC SHALL deactivate the local link.

5.5 Connectionless Transport Mode Procedures

Connectionless transport mode does not require any prior connection establishment, and consequently does not require disconnection. Once a service access point has been enabled in the LLC, information may be sent to, or received from, the remote LLC SAP.

5.5.1 Logical Data Link Parameters

5.5.1.1 Maximum Information Unit Size for UI PDUs

The information field of a UI PDU sent by the local LLC SHALL NOT be greater than the remote LLC’s Link MIU.

5.5.2 Information Transfer

Information transfer SHALL be accomplished by sending UI PDUs and the local LLC SHALL be able to send or receive UI PDUs at any time. Sending or receiving UI PDUs SHALL NOT affect any state variable associated with any data link connection.

Reception of UI PDUs SHALL NOT be acknowledged by the receiving LLC.

The MIU values used in connectionless mode SHALL be the Link MIU determined according to section 5.2.3.

Local conditions at the receiving LLC may result in the discarding of valid UI PDUs. It is the choice of an implementation whether or not to report discarded UI PDUs to the service user.

5.6 Connection-oriented Transport Mode Procedures

5.6.1 Data Link Connection State Variables

For the purpose of this specification, each LLC can be said to have a virtual state for each data link connection. The following normative statements with respect to these state variables set forth the operational requirements of the LLC but do not mandate any particular implementation in either hardware or software.

In the following descriptions, the term *modulo-16* SHALL denote the inclusive range of values between 0 and 15, with the numbers cycling through the entire range. Addition and subtraction on these values SHALL conform to the rules of modulo arithmetic. For example, incrementing a state variable by one, when it's present value is 15, SHALL yield a new value 0.

5.6.1.1 Send State Variable V(S)

The send state variable V(S) SHALL denote the sequence number, modulo-16, of the next in-sequence I PDU to be sent on a specific data link connection. The value of the send state variable V(S) SHALL be incremented by one following each successive I PDU transmission on the associated data link connection. V(S) SHALL exhibit the invariant condition that it is always less than or equal to $(V(SA) + RW(R)) \text{ modulo-16}$.

5.6.1.2 Send Acknowledgement State Variable V(SA)

The send acknowledgement state variable V(SA) SHALL denote the most recently received N(R) value for a specific data link connection.

5.6.1.3 Receive State Variable V(R)

The receive state variable V(R) SHALL denote the sequence number, modulo-16, of the next in-sequence I PDU to be received on a specific data link connection. The value of the receive state variable V(R) associated with a specific data link connection SHALL be incremented by one whenever an error-free, in-sequence I PDU is received whose send sequence number N(S) equals the current value of the receive state variable V(R) for the data link connection.

5.6.1.4 Receive Acknowledgement State Variable V(RA)

The receive acknowledgement state variable V(RA) SHALL denote the most recently sent N(R) value for a specific data link connection.

5.6.2 Data Link Connection Parameters

5.6.2.1 Maximum Information Unit Size for I PDUs

The maximum information unit size for I PDUs SHALL denote the maximum number of octets in the information field of any I PDU exchanged on the data link connection. The local and remote LLCs each establish and maintain distinct MIU values for each data link connection endpoint. In this way each data link connection has two MIU values: these are the local MIU and the remote MIU.

The default value of the MIU for I PDUs SHALL be 128 octets.

An LLC MAY announce a larger MIU for a data link connection by transmitting an MIUX parameter within the information field of the CONNECT or CC PDU that it sends to the remote LLC. The announced MIU SHALL NOT exceed the local LLC's Link MIU value.

If an LLC receives an MIUX parameter, it SHALL establish the remote MIU value as defined in section 4.5.2.

An LLC SHALL NOT send an I PDU with an information field larger than the remote MIU established for the data link connection.

If an LLC receives an I PDU with an information field larger than the local MIU established for the data link connection it SHALL respond as defined in section 5.6.4.5.

5.6.2.2 Local Receive Window Size, RW(L)

The local receive window size SHALL denote the maximum number of sequentially numbered I PDUs that the local LLC allows the remote LLC to have unacknowledged. The local receive window size SHALL be in the inclusive range of values between 1 and 15. It MAY be sent by the local LLC with a CONNECT PDU or CC PDU.

5.6.2.3 Remote Receive Window Size, RW(R)

The remote receive window size SHALL denote the maximum number of sequentially numbered I PDUs that the remote LLC allows the local LLC to have unacknowledged. It MAY be received within a CONNECT or CC PDU.

5.6.3 Connection Establishment

Either LLC MAY be able to establish a data link connection.

The LLC instructed to establish a data link connection SHALL send the CONNECT PDU. The CONNECT PDU MAY provide connection specific parameters.

The local LLC SHALL NOT send more than one CONNECT PDU with the same SSAP address value before receipt of a CC or DM PDU with the DSAP value equal to the SSAP value in the pending CONNECT PDU. Note that the CC PDU may contain a SSAP value that is different from the DSAP value in the CONNECT PDU.

If the LLC receives a CC PDU with a DSAP value equal to the SSAP value of a sent but not yet acknowledged CONNECT PDU, it SHALL act as follows:

1. Set the state variables V(S) and V(R) and V(SA) and V(RA) to 0
1. Process connection specific parameters in the information field of the CC PDU
2. Enter the information transfer phase described in section 5.6.4.

If the LLC receives a DM PDU with a DSAP value equal to the SSAP value of a sent but not yet acknowledged CONNECT PDU, it SHALL abandon connection establishment and report the reason to the service layer.

If the local LLC receives a CONNECT PDU and is unable to process the connection request, it SHALL return a DM PDU with the appropriate reason code (cf. table 3 in section 4.3.8) to the remote LLC at the earliest opportunity.

Otherwise, the local LLC SHALL act as follows:

1. Process connection specific parameters (if provided with the CONNECT PDU)
2. Indicate to the service layer that a data link connection is requested from the remote LLC
3. Await notification from the service layer to either accept or reject the connection request
4. If the service layer rejects the connection request, send a DM PDU with the appropriate reason code to the remote LLC
5. If the service layer accepts the connection request, send a CC PDU to the remote LLC.
6. Set the state variables $V(S)$ and $V(R)$ and $V(SA)$ and $V(RA)$ to 0
7. Enter the information transfer phase described in section 5.6.4.

5.6.4 Information Transfer

During the information transfer phase, the LLC SHALL be able to send and receive information according to the procedures described in this section.

5.6.4.1 Sending I PDUs

While the send state variable $V(S)$ is equal to the send acknowledge state variable $V(SA)$ plus the remote receive window size $RW(R)$, the LLC SHALL NOT send an I PDU on that data link connection.

When the LLC sends an I PDU, it SHALL set the $N(S)$ field equal to the current value of the send state variable $V(S)$, and set the $N(R)$ field equal to the current value of the receive state variable $V(R)$ for that data link connection. After sending the I PDU, the LLC SHALL increment the send state variable $V(S)$ by one.

5.6.4.2 Receiving I PDUs

When an I PDU is received with the send sequence number $N(S)$ equal to the receive state variable $V(R)$, the LLC SHALL pass the service data unit, contained in the information field, to the service access point and increment by one its receive state variable, $V(R)$. The LLC SHALL then acknowledge receipt of that I PDU as described in section 5.6.4.3.

5.6.4.3 Sending Acknowledgement

If at least one I PDU is available to send on this data link connection, the received I PDU will be acknowledged when sending the next I PDU.

If no I PDU is available to send on this data link connection, and the data link connection is not presently in the receiver busy condition, the LLC SHALL send an RR PDU.

If no I PDU is available to send on this data link connection, and the data link connection is presently in the receiver busy condition, the LLC SHALL send an RNR PDU.

5.6.4.4 Receiving Acknowledgement

The sequence field of a numbered PDU (i.e. RR, RNR or I) contains the receive sequence number $N(R)$. This $N(R)$ field value indicates that I PDUs numbered up through $N(R)-1$ have been received correctly and SHALL be considered as acknowledged.

This $N(R)$ value becomes the new value of the send acknowledgement state variable, $V(SA)$, for this data link connection.

5.6.4.5 Reception of erroneous PDUs

If an erroneous PDU is received on a data link connection, the local LLC SHALL send the FRMR PDU. A PDU with one or more of the following conditions SHALL be considered erroneous and the W, I, R and S flags SHALL indicate the error condition as defined in table 7:

Table 7: FRMR Error Conditions

Error Condition	W Flag	I Flag	R Flag	S Flag
A PDU with a PTYPE that is reserved (undefined)	1	0	0	0
A PDU with an information field that is not permitted for this PDU TYPE	1	1	0	0
An I PDU with an information field exceeding the maximum information field length for that data link connection for reception at the rejecting LLC.	0	1	0	0
A numbered PDU with an N(R) field value where repeatedly incrementing this N(R) value, modulo-16, results in a value that would become equal to V(SA) before it would become equal to V(S)	0	0	1	0
A numbered PDU with an N(S) that is greater than or equal to $(V(RA) + RW(L))$, modulo-16, for that data link connection	0	0	0	1
A numbered PDU with an N(S) that is not an increment by 1 of the last received N(S)	0	0	0	1

If an erroneous PDU exhibits more than one error condition, the flag values SHALL be the logical OR of the flag values associated with the individual error conditions in table 7. An LLC receiving an FRMR PDU on a data link connection can acquire further knowledge about the error conditions causing the FRMR exception by inspecting the additional fields provided with the FRMR PDU.

After sending or receiving a FRMR PDU the local LLC SHALL pass a disconnect indication to the service access point for that data link connection. The data link connection SHALL be closed and resources released. The service layer is responsible for initiating any corrective action by for example instructing the LLC to re-establish the data link connection.

5.6.5 Receiver Busy Condition

While in the information transfer phase a condition could arise at the service access point that prevents acceptance of service data units from the local LLC. An LLC SHALL send an RNR PDU on a data link connection when it is temporarily unable to dispatch service data units to the local SAP. While in the receiver busy condition, the local LLC SHALL continue to accept and buffer I PDUs, process control requests and transmit I PDUs and control requests. When the receiver busy condition clears, the local LLC SHALL send an RR PDU on the data link connection.

When sending the RNR PDU the LLC SHALL set the N(R) field to the current value of the receive state variable V(R).

5.6.5.1 Receiving an RNR PDU

An LLC receiving an RNR PDU on a data link connection SHALL cause that data link connection to enter the remote receiver busy condition. While in this condition the local LLC SHOULD stop sending I PDUs on that data link connection.

5.6.5.2 Receiving an RR PDU

An LLC receiving an RR PDU on a data link connection which is in the remote receiver busy condition SHALL clear the remote receiver busy condition and resume the normal information transfer phase.

5.6.6 Connection Termination

When either LLC wishes to terminate the data link connection, it SHALL send the DISC PDU and enter the disconnecting phase.

While in the disconnecting phase the LLC SHALL NOT send further PDUs on that data link connection and SHALL discard subsequently received PDUs other than DM. Upon receipt of a DM PDU, the LLC SHALL pass a disconnect indication to the service access point for that data link connection. The data link connection SHALL be closed and resources released. The LLC SHALL discard subsequently received information PDUs.

When receiving a DISC PDU, the LLC SHALL return a DM PDU and pass a disconnect indication to the service access point for that data link connection. The data link connection SHALL then be closed.

5.7 Aggregation Procedure

An LLC SHALL be able to receive and disaggregate AGF PDUs. An LLC SHOULD be able to aggregate and send AGF PDUs.

AGF PDUs SHALL exclusively be used by the LLC Link Management component.

The LLC Link Management component SHALL aggregate PDUs in the same order as they would have been transmitted without aggregation.

The LLC Link Management component SHALL dispatch aggregated PDUs in the same order as they appear in the AGF PDU.

Reception of AGF PDUs SHALL NOT be acknowledged by the receiving LLC.

5.8 Symmetry Procedure

The symmetry procedure provides the appearance of asynchronous balanced mode (ABM) communication on top of MAC layers operating in normal response mode (NRM) and enables link loss detection. By performing the symmetry procedure, the peers mutually pass permission to send back and forth. The link timeout (LTO) parameter allows each peer to determine when communication over the LLCP Link has become impossible.

After receipt of an LLC PDU, the receiving LLC SHALL commence sending an LLC PDU to the remote LLC within the time interval determined by the link timeout (LTO) parameter of the local LLC. In case no other PDU can be sent, the LLC SHALL send a SYMM PDU prior to the expiration of the LTO of the local LLC.

After sending an LLC PDU, if reception of a PDU from the remote LLC does not commence within the time interval determined by the remote LLC's link timeout (LTO) parameter, the local LLC SHALL execute the LLC link deactivation procedure (cf. section 5.4).

Compliance to the symmetry procedure SHALL be verified by observing the communication traffic between the peer devices.

6 MAC Mappings

The LLC layer can be mapped on top of different MAC layers, as long as they comply with the requirements set forth in section 6.1. The MAC layers defined with this version of the LLC specification are given thereafter, starting with section 6.2.

6.1 Requirements

The LLC protocol places the following mandatory requirements on the medium access control (MAC) layer. All MAC layers enabled with this specification **MUST** support the complete set of requirements either directly or by the mapping procedures defined in this specification. These requirements are:

1. Provide framing of LLC PDUs. Segmentation and reassembly procedures on the MAC layer PDU exchange **MUST NOT** appear visible on the LLC layer.
2. Accept any defined LLC PDU from the LLC layer in their entirety.
3. Pass any defined LLC PDU to the LLC layer in their entirety and provide the length of the LLC PDU.
4. Transmit any defined LLC PDU without error and in sequence. Transmission errors **MUST** be detected and corrected before an LLC PDU is passed to the LLC layer.
5. Indicate unrecoverable transmission errors to the LLC layer.

6.2 Definition of the ISO18092 MAC Mapping

The MAC layer defined within this section is based on the ISO/IEC 18092 [DIGIPROT] specification. Elements and procedures required by the LLC layer, and not described in [DIGIPROT], are defined in this section. The terminology used to define the MAC layer is oriented towards the LLC layer service user; it is not intended to define a MAC layer usable for other purposes.

6.2.1 Restrictions on the use of ISO 18092

The following restrictions on the use of the ISO/IEC 18092 specification are made for the ISO/IEC 18092 MAC layer defined with this specification:

1. The DID feature **SHALL NOT** be used.
2. The NAD feature **SHALL NOT** be used.
3. Frame waiting time extensions (WTX) **SHALL NOT** be used.
4. Except as required by the Digital Protocol Specification [DIGIPROT], the Target Presence procedure **SHALL NOT** be used and the Attention command **SHALL NOT** be sent.
5. The Length Reduction values L_{Ri} and L_{Rt} **MUST** be 11b.
6. For the Target device, the scaled value of RWT **MUST** be less than or equal to the scaled value of the LLC Link Timeout (LTO).

6.2.2 Restrictions on LLC when using ISO 18092

The following restrictions apply to an LLC operating over an ISO/IEC 18092 MAC layer as defined with this specification:

1. For the Target device, the value of LTO SHOULD be sufficiently larger than RWT to allow one or more MAC error recovery cycles.
2. For the Target device, after receipt of an LLC PDU, the receiving LLC SHALL commence sending an LLC PDU to the remote LLC within the time interval determined by the scaled value of RWT. In case no other PDU can be sent, the LLC SHALL send a SYMM PDU prior to the expiration of RWT.

6.2.3 MAC Link Activation

The ISO/IEC18092 MAC component SHALL use the three octet sequence “46h 66h 6Dh” as the NFC Forum LLC magic number. This magic number is encoded into the ATR_REQ / ATR_RES General Bytes fields, as described below. The use of the magic number by the Initiator and Target SHALL indicate compatibility with the requirements of this specification.

The link activation phase SHALL be started when a peer device capable of executing the LLC peer-to-peer protocol enters communication range, and the local device is instructed to perform peer-to-peer communication. The link activation phase is different for the Initiator and the Target device and is described separately for each role.

6.2.3.1 Link Activation procedure for the ISO 18092 Initiator

Following the initialization and anti-collision procedure defined in [DIGIPROT], the Initiator device sends the Attribute Request ATR_REQ command. The format of the ATR_REQ SHALL be as shown in figure 25.

The Initiator SHALL include the NFC Forum LLC magic number in the first three octets of the ATR_REQ General Bytes field.

All LLC parameters defined in section 4.5 table 5 for use in PAX PDUs that are to be exchanged MUST be included as TLVs beginning at the fourth octet of the ATR_REQ General Bytes field. The PAX PDU exchange described in the LLC link activation procedure (cf. section 5.2) SHALL NOT be used.

The ATR_REQ General Bytes field SHALL NOT contain any additional information.

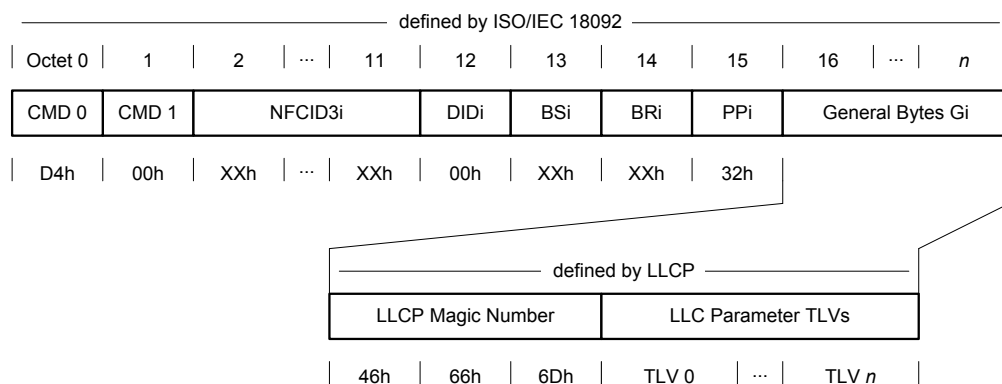


Figure 25: Format of the Attribute Request (ATR_REQ)

Upon receipt of the Attribute Response ATR_RES the Initiator SHALL verify that the first three octets of the General Bytes field are equal to the NFC Forum LLC magic number defined in section 6.2.2. If the octets are equal to the NFC Forum LLC magic number, the Initiator SHALL notify the local LLC component about the MAC link activation completion and SHALL then enter normal operation described in chapter 6.2.5.

If the first three octets of the General Bytes field are not equal to the NFC Forum LLC magic number, the link activation SHALL fail. In this case, any further communication between the Initiator and the Target is out of scope of this specification and left to implementation.

6.2.3.2 Link Activation procedure for the ISO 18092 Target

Following the initialization and anti-collision procedure defined in [DIGIPROT], the Target device waits until the receipt of the Attribute Request ATR_REQ command.

Upon receipt of ATR_REQ the Target SHALL verify that the first three octets of the General Bytes field are equal to the NFC Forum LLC magic number defined in section 6.2.2. If the octet sequence is equal to the NFC Forum LLC magic number, the Target responds by sending the Attribute Response ATR_RES, as defined in [DIGIPROT]. The format of the ATR_RES SHALL be as shown in figure 26.

The Target SHALL include the NFC Forum LLC magic number in the first three octets of the ATR_RES General Bytes field.

All LLC parameters defined in section 4.5 table 5 for use in PAX PDUs that are to be exchanged MUST be included as TLVs beginning at the fourth octet of the ATR_RES General Bytes field. The PAX PDU exchange described in the LLC link activation procedure (cf. section 5.2) SHALL NOT be used.

The ATR_RES General Bytes field SHALL NOT contain any additional information.

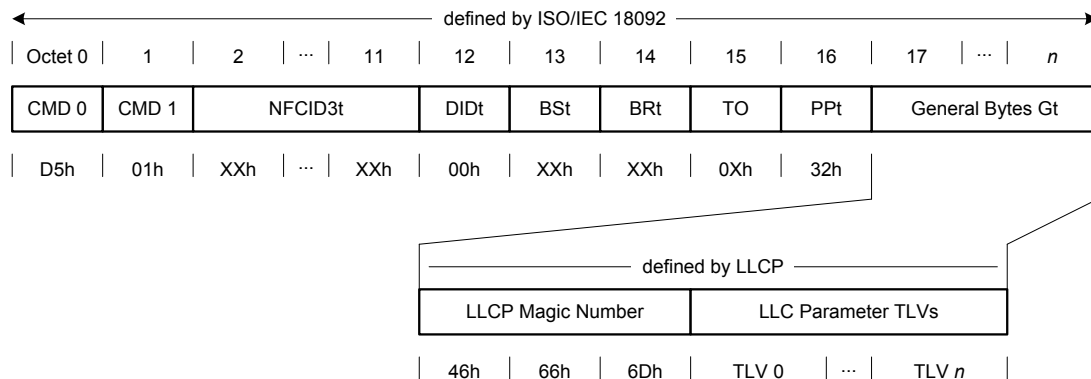


Figure 26: Format of the Attribute Response (ATR_RES)

After sending ATR_RES the Target SHALL notify the local LLC component about the MAC link activation completion and SHALL then enter normal operation described in section 6.2.5.

If the magic number in the received ATR_REQ cannot be verified, the link activation SHALL fail. In this case, any further communication between the Target and the Initiator is out of scope of this specification and left to implementation.

6.2.4 MAC Link Deactivation

An 18092 MAC SHALL perform link deactivation either if requested by the local LLC or if communication with the peer MAC has become impossible and error recovery as described by [DIGIPROT] did not re-establish communication.

The MAC performing the Initiator role SHALL:

- On request of the local LLC, stop sending subsequent ISO/IEC 18092 DEP_REQ commands, send the Deselect Request (DSL_REQ) as specified in [DIGIPROT].
- On the condition of unsuccessful error recovery, send the Deselect Request (DSL_REQ) as specified in [DIGIPROT] and indicate link failure to the local LLC.

The MAC performing the Target role SHALL:

- On request of the local LLC, stop responding to ISO/IEC 18092 DEP_REQ commands until the peer device sends the Deselect Request (DSL_REQ), and confirm link deactivation back to the local LLC.
- On the condition that the local device has received the Deselect Request (DSL_REQ), send the Deselect Response (DSL_RES) as specified in [DIGIPROT] and indicate link deactivation to the local LLC.
- On the condition that the peer device has stopped generating the RF field, indicate link deactivation to the local LLC.

6.2.5 Normal Operation

In normal operation mode, both the Initiator and the Target MAC component SHALL be able to accept LLC PDUs from the local LLC and SHALL be able to pass LLC PDUs to the local LLC.

6.2.5.1 Transmission of LLC PDUs

LLC PDUs SHALL be transmitted using the ISO/IEC 18092 Data Exchange Protocol Request (DEP_REQ) and Response (DEP_RES) protocol frames. Outbound LLC PDUs SHALL be mapped onto consecutive transport data bytes of DEP_REQ or DEP_RES protocol frames. If an LLC PDU does not fit completely into a single DEP_REQ or DEP_RES protocol frame, the chaining procedure of [ISO18092] SHALL be used to transmit the LLC PDU in a number of consecutive segments; when chaining is activated, every segment except the last SHALL use the maximum number of transport data bytes.

6.2.5.2 Reception of LLC PDUs

LLC PDUs SHALL be passed to the local LLC when the last octet has been received without transmission errors. LLC PDUs transmitted in segments MUST be assembled by the receiving MAC. With indication to the local LLC, the total number of octets of the received LLC PDU MUST be made available.

6.2.6 Error Handling

If error recovery as described by [DIGIPROT] does not re-establish communication, the ISO/IEC 18092 MAC SHALL trigger the MAC link deactivation procedure described in section 6.2.4.

A. Revision History

The following table outlines the revision history of Logical Link Control Protocol.

Table 8: Revision History

Document Name	Revision and Release Date	Status	Change Notice	Supersedes
Logical Link Control Protocol	1.0, December 11, 2009	Final		All previous versions



Connection Handover

Technical Specification

NFC Forum™

Connection Handover 1.2

NFCForum-TS-ConnectionHandover_1_2.doc

2010-07-07

RESTRICTIONS ON USE

This specification is copyright © 2005-2010 by the NFC Forum, and was made available pursuant to a license agreement entered into between the recipient (Licensee) and NFC Forum, Inc. (Licensor) and may be used only by Licensee, and in compliance with the terms of that license agreement (License). If you are not the Licensee, you are not authorized to make any use of this specification. However, you may obtain a copy at the following page of Licensor's Website: http://www.nfc-forum.org/specs/spec_license after entering into and agreeing to such license terms as Licensor is then requiring. On the date that this specification was downloaded by Licensee, those terms were as follows:

1. LICENSE GRANT.

Licensor hereby grants Licensee the right, without charge, to copy (for internal purposes only) and share the Specification with Licensee's members, employees and consultants (as appropriate). This license grant does not include the right to sublicense, modify or create derivative works based upon the Specification.

2. NO WARRANTIES.

THE SPECIFICATION IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY, COMPLETENESS AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL LICENSOR, ITS MEMBERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE SPECIFICATION.

3. THIRD PARTY RIGHTS.

Without limiting the generality of Section 2 above, LICENSOR ASSUMES NO RESPONSIBILITY TO COMPILE, CONFIRM, UPDATE OR MAKE PUBLIC ANY THIRD PARTY ASSERTIONS OF PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS THAT MIGHT NOW OR IN THE FUTURE BE INFRINGED BY AN IMPLEMENTATION OF THE SPECIFICATION IN ITS CURRENT, OR IN ANY FUTURE FORM. IF ANY SUCH RIGHTS ARE DESCRIBED ON THE SPECIFICATION, LICENSOR TAKES NO POSITION AS TO THE VALIDITY OR INVALIDITY OF SUCH ASSERTIONS, OR THAT ALL SUCH ASSERTIONS THAT HAVE OR MAY BE MADE ARE SO LISTED.

4. TERMINATION OF LICENSE.

In the event of a breach of this Agreement by Licensee or any of its employees or members, Licensor shall give Licensee written notice and an opportunity to cure. If the breach is not cured within thirty (30) days after written notice, or if the breach is of a nature that cannot be cured, then Licensor may immediately or thereafter terminate the licenses granted in this Agreement.

5. MISCELLANEOUS.

All notices required under this Agreement shall be in writing, and shall be deemed effective five days from deposit in the mails. Notices and correspondence to the NFC Forum address as it appears below. This Agreement shall be construed and interpreted under the internal laws of the United States and the Commonwealth of Massachusetts, without giving effect to its principles of conflict of law.

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

Contents

1	Introduction.....	1
1.1	Objectives.....	1
1.2	Purpose.....	1
1.3	Mission Statement and Goals.....	1
1.4	References.....	2
1.5	Administration.....	2
1.6	Special Word Usage.....	2
1.7	Name and Logo Usage.....	3
1.8	Intellectual Property.....	3
1.9	Glossary.....	3
2	Handover Protocol	5
2.1	Introduction.....	5
2.2	Negotiated Handover.....	5
2.3	Static Handover.....	8
2.4	Message Composition.....	8
2.5	Carrier Type Identification.....	9
2.6	Carrier Power State.....	9
2.7	Handover Request Collision.....	10
2.8	Message Transport.....	11
2.9	Version Handling.....	11
2.10	Security Considerations.....	12
3	NDEF Structure.....	13
3.1	Message Definitions.....	13
3.1.1	Handover Request Message.....	13
3.1.2	Handover Select Message.....	13
3.2	Global Record Definitions.....	14
3.2.1	Handover Request Record.....	14
3.2.2	Handover Select Record.....	15
3.2.3	Handover Carrier Record.....	16
3.3	Local Record Definitions.....	17
3.3.1	Alternative Carrier Record.....	17
3.3.2	Collision Resolution Record.....	18
3.3.3	Error Record.....	19
A.	ABNF Definition of Handover Messages and Records	21
B.	Revision History	23

Figures

Figure 1: Negotiated Handover with Single Selection	5
Figure 2: Negotiated Handover with Multiple Selections	6
Figure 3: Negotiated Handover with Multiple Selections	6
Figure 4: Negotiated Handover with a Power-constrained Device	7
Figure 5: Negotiated Handover Message Sequence	7
Figure 6: Static Handover.....	8
Figure 7: General Message Composition	9
Figure 8: Handover Request Message Structure	13
Figure 9: Handover Select Message Structure	13
Figure 10: Payload of the Handover Request Record	14
Figure 11: Payload of the Handover Select Record	15
Figure 12: Handover Carrier Record Encoding.....	16
Figure 13: Alternative Carrier Record Layout	17
Figure 14: Carrier Data Reference Encoding	18
Figure 15: Auxiliary Data Reference Encoding	18
Figure 16: Payload of the Collision Resolution Record	19
Figure 17: Payload of the Error Record	19

Tables

Table 1: Allowed CTF Field Values	16
Table 2: Carrier Power State Values	17
Table 3: Error Reason Values.....	19
Table 4: Contents of the Error Data Field	20
Table 5: Revision History.....	23

1 Introduction

The Connection Handover specification defines NFC Forum Well Known Types and the corresponding message structure that allows negotiation and activation of an alternative communication carrier. The negotiated communication carrier would then be used to perform certain activities between the two devices, such as printing to a Bluetooth printer or streaming video to a WLAN television set.

1.1 Objectives

The objective of this specification is to provide a generic mechanism to negotiate and activate an alternative communication carrier between two NFC Forum devices. The mechanism is intended to allow for applications to be built that need to use a communication system other than NFC to transmit relatively large amounts of data or use services not available via the NFC link.

It is not the objective of this specification to provide any definitions that are specific to alternative communication carriers or to define how an application would accomplish a given task once switched to an alternative carrier. Such information needs to be given in other specifications, either released in the future by the NFC Forum or disclosed by other organizations.

1.2 Purpose

The Connection Handover specification aims to enable applications to take advantage of NFC technology for initiating and executing user-defined activities between devices. This specification delivers a toolset that ensures compatibility between the actors, but requires further definitions to achieve full interoperability for specific alternative carrier technologies.

1.3 Mission Statement and Goals

Near Field Communication (NFC) technology can be used to design extremely intuitive user interfaces that involve activities between two devices. For example, a digital still picture camera might directly print an image that is currently shown in review mode when the user touches the camera's NFC interface to an NFC-equipped printer. Likewise, music files could be automatically synchronized between a mobile player and a home media center with a simple touch.

However, NFC alone might not be suitable for some scenarios, such as transferring large files, due to the inconvenience of keeping the interfaces of the two devices in close proximity for an extended period of time. Furthermore, many existing applications are designed to use other communication carriers and cannot be easily modified to use NFC.

The Connection Handover specification intends to solve these issues by providing the means for two NFC-equipped devices to negotiate and use an alternative communication carrier that is better suited to perform the desired task.

1.4 References

- [LLCP] “NFC Logical Link Control Protocol (LLCP) Technical Specification”, NFC Forum, 2009
- [NDEF] “NFC Data Exchange Format (NDEF) Technical Specification”, NFC Forum, 2006.
- [NFC RTD] “NFC Record Type Definition (RTD) Technical Specification”, NFC Forum, 2006.
- [RFC 2046] N. Freed, N. Borenstein, “Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types” RFC 2046, Innosoft, First Virtual, November 1996.
- [RFC 2119] S. Bradner, “Key words for use in RFCs to Indicate Requirement Levels”, RFC 2119, Harvard University, March 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.
- [RFC 4234] D. Crocker, P. Overell, “Augmented BNF for Syntax Specifications: ABNF”, RFC 5234, January 2008.

1.5 Administration

The NFC Forum Connection Handover 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/>

The Reference Applications Framework Technical Working Group maintains this specification.

1.6 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.

1.7 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.

1.8 Intellectual Property

The Connection Handover specification conforms to the Intellectual Property guidelines specified in the NFC Forum's Intellectual Property Rights Policy, as approved on November 9, 2004 and outlined in the NFC Forum Rules of Procedures, as approved on December 17, 2004, and revised on June 23, 2007 and August 4, 2009.

1.9 Glossary

Alternative Carrier

A (wireless) communication technology that can be used for data transfers between a Handover Requester and a Handover Selector.

Carrier Configuration Data

The information needed to connect to an alternative carrier. The exact amount of information depends on the carrier technology.

Handover Requester

An NFC Forum Device that begins the Handover Protocol by issuing a Handover Request Message to another NFC Forum Device.

Handover Selector

An NFC Forum Device that constructs and replies to a Handover Select Message as a result of a previously received Handover Request Message or an NFC Forum Tag that provides a pre-set Handover Select Message for reading.

Negotiated Handover

An exchange of NDEF messages that allows two NFC Forum Devices to agree on a (set of) alternative carrier(s) to be used for further data exchange.

Static Handover

Provision of an NDEF message on an NFC Forum Tag that allows a reading NFC Forum Device to select and use alternative carriers for further data exchange.

2 Handover Protocol

2.1 Introduction

This specification defines NDEF messages that enable a Handover Requester to negotiate an alternative communication carrier with a Handover Selector over the NFC link. As a special case, it also enables a Handover Requester to retrieve the possible alternative communication carrier(s) from an NFC Forum Tag, but this has some limitations due to the static nature of information stored on a Tag. The first case is called “Negotiated Handover” and is described in section 2.2, while the second case is called “Static Handover” and is described in section 2.3.

The Handover Requester, in the scope of this specification, is defined to be the device that initiates the handover operation. The Handover Selector device is defined to be the device that is initially passive and that responds to the Handover message. The Handover Selector does not start any activity such as generating a handover message.

2.2 Negotiated Handover

Negotiated Handover allows two devices to negotiate one or more alternative carriers for further data exchange. The exemplary use case shown in Figure 1 illustrates how a Handover Requester uses the embedded NFC Forum Device to exchange connection handover information with the Handover Selector to finally select a matching alternative carrier. In the example, the application running on the Handover Requester first announces its alternative carriers (Wi-Fi and Bluetooth wireless technology) to the Handover Selector, and then receives a carrier selection (Bluetooth wireless technology) as the only choice, and finally performs Bluetooth pairing and data exchange.

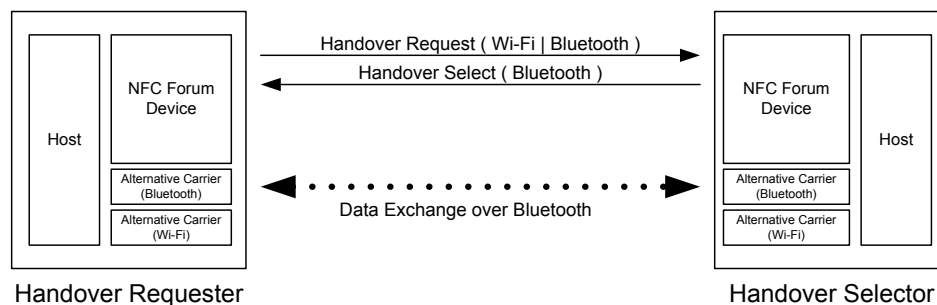


Figure 1: Negotiated Handover with Single Selection

If the Handover Selector supports multiple alternative carriers, it might return more than one selection (see Figure 2). In this case, the Handover Requester is free to choose any of the returned carriers or even try to simultaneously connect to more than one alternative carrier. However, if the Handover Requester attempts to choose one of the selected carriers, it should interpret the order that they are listed in the Handover Select message as a preference indication. In the example of Figure 2, the Handover Requester has decided against the Handover Selector’s preference for Wi-Fi and has used Bluetooth wireless technology instead.

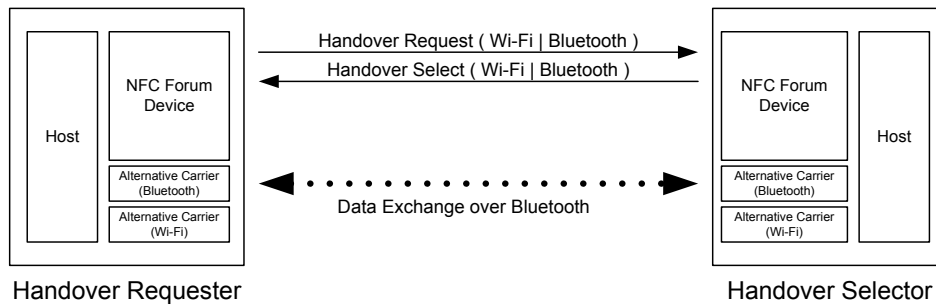


Figure 2: Negotiated Handover with Multiple Selections

A third example (Figure 3) illustrates how a Handover Requester might use a second alternative carrier selection to enhance the user experience. In the example, the Handover Selector again returns both Wi-Fi and Bluetooth wireless technology, but this time with Bluetooth wireless technology as the first preference. The Requester first tries to establish a Bluetooth connection that fails (for example, because the devices have moved outside of Bluetooth radio range). In this case, the application might decide to try the Wi-Fi connection before aborting.

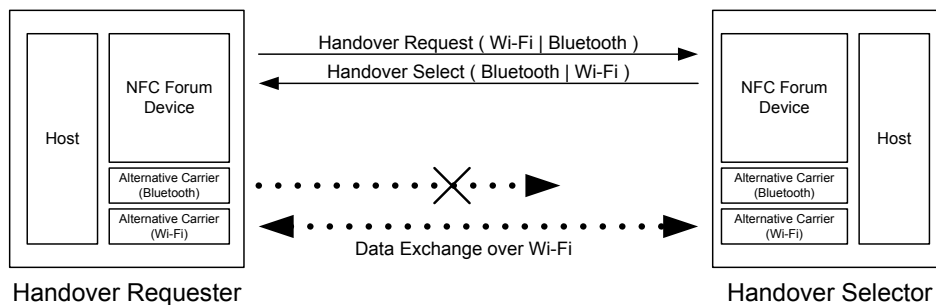


Figure 3: Negotiated Handover with Multiple Selections

A Handover Selector device with limited power resources (for example, battery driven) might not want to activate all alternative carrier circuits in case it has more than a single carrier in common with the Handover Requester. In this situation, the Handover Selector MAY set the Carrier Power State flag of all returned Alternative Carrier Records to zero (inactive). This causes the Handover Requester to send a subsequent Handover Request Message with only one of the previously received alternative carriers listed. The Handover Selector SHALL acknowledge this request with the return of a Handover Select Message (see Figure 4), where the Carrier Power State flag MUST be set to 1 (active). Note that the Handover Selector does not need to maintain state information between its two responses.

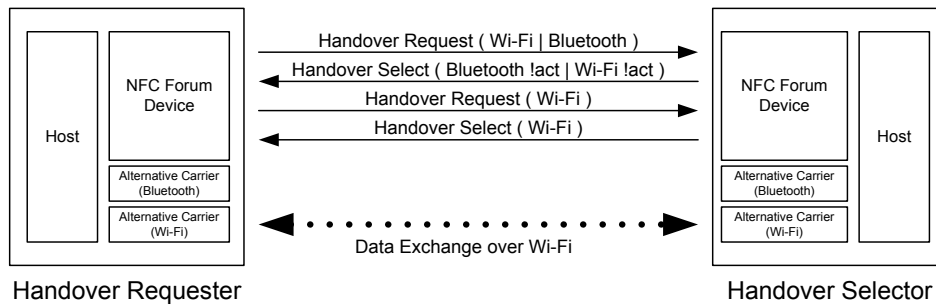


Figure 4: Negotiated Handover with a Power-constrained Device

As shown with the preceding examples, the process of a negotiated handover is performed with the exchange of two messages in the following sequence (see Figure 5):

1. The Handover Requester sends a Handover Request Message to the Handover Selector device. This message proposes a number of alternative communication carriers that the Handover Requester would be able to use for the intended activity. A proposed carrier might be provided with or without carrier configuration information. The message might state further carrier-associated requirements.
2. The Handover Selector device compares this list with its own communication facilities, possibly taking connectivity status and carrier specific requirements into account. It then returns a Handover Select Message containing a list of appropriate communication carriers, each associated with a carrier-specific configuration record. If it is not possible to use any of the proposed carriers, the Handover Selector returns an empty list.

If the Handover Selector device returns a non-empty list of alternative carriers, the handover protocol is successfully completed and establishing communication depends on the selected carrier(s). Information indicating how to establish a connection to the alternative carrier(s) **MUST** be provided in a carrier-specific configuration record inside the Handover Select Message and this record **MUST** be referenced as described in section 2.4. If the Handover Selector accepts a carrier that has been proposed with configuration information, it needs to copy the relevant information to the corresponding Handover Select Message.

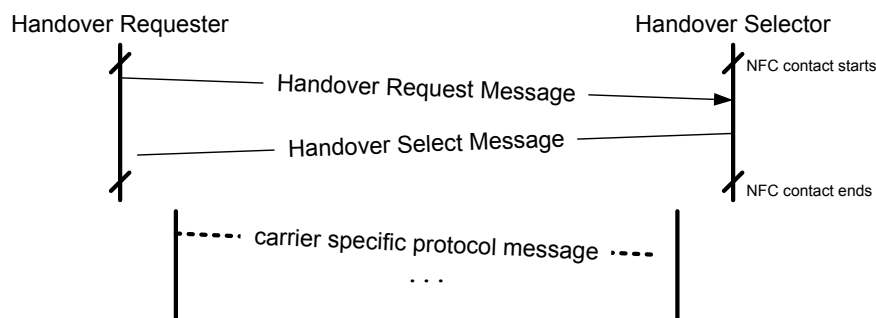


Figure 5: Negotiated Handover Message Sequence

If the Handover Selector device does not acknowledge at least one of the proposed alternative carriers, the Handover Requester might want to repeat the sequence with different settings. This

could be useful if the Handover Requester has a strong preference on a certain communication carrier or properties. However, the Handover Selector device **SHOULD** compare the proposed carriers in the same order as they are listed in the Handover Request Message and give preference to those listed first.

A Handover Selector **SHOULD** answer a Handover Request Message within 1 second; otherwise, the Handover Requester **MAY** assume a processing error. Note that the message transport layer is assumed to guarantee delivery of Handover messages over the NFC link. See section 2.8 for more details.

2.3 Static Handover

The Static Handover can be used when the Handover Selector device does not constitute an NFC Forum Device but has a (cheaper) NFC Forum Tag attached (see Figure 6). NFC Forum Tags provide storage space that can be read or written but might not have an internal connection with the Host processor. It is not possible for an NFC Forum Tag to receive and interpret a Handover Request Message and to dynamically construct a corresponding Handover Select Message.

In this case, the Tag contains a Handover Select Record plus additional information records that are referenced as described in section 2.4. Because this is static information, it cannot be adapted to any possible requirements that a Handover Requester might provide with Negotiated Handover. Also, dynamic information (for example, a dynamically-assigned IP address) cannot be provided sensibly.

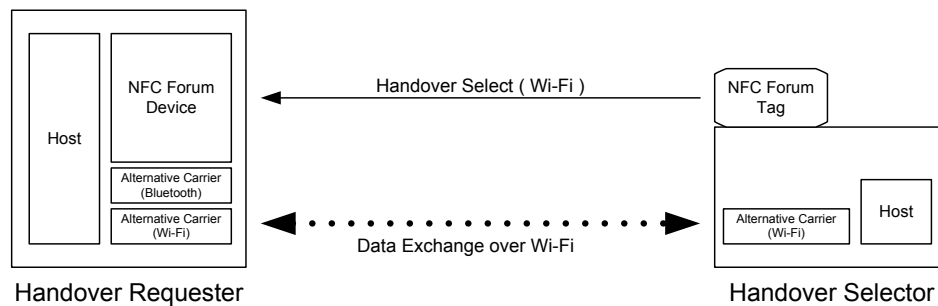


Figure 6: Static Handover

A further drawback of Static Handover is that the Handover Selector's Host processor will not be able to activate its alternative carrier circuits as part of the handover process. All carrier circuits either have to be continuously active or have to be activated explicitly by the user.

2.4 Message Composition

A handover message is composed of either a Handover Request Record (NFC Forum Global Type "Hr") or a Handover Select Record (NFC Forum Global Type "Hs"), followed by an arbitrary number of other NDEF records.

Within a Handover Request or Handover Select Record, a sequence of Alternative Carrier Records (NFC Forum Local Type "ac") defines the alternative carriers that are requested or selected, respectively. The actual information is provided with the NDEF records that follow the leading message record, and the Alternative Carrier Record provides references to those belonging to the given alternative carrier (see Figure 7).

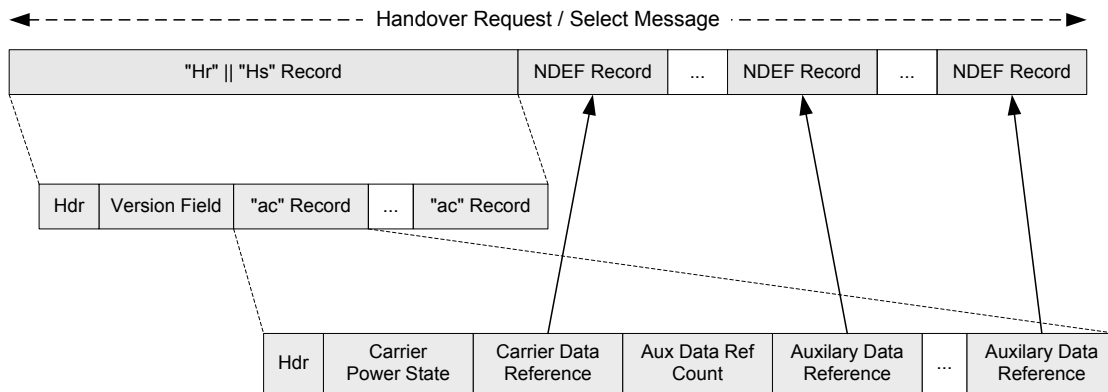


Figure 7: General Message Composition

The record references are established using the URI-based Payload Identification mechanism described in the NDEF specification [NDEF]. The URI reference values SHALL be encoded as relative URIs with the virtual base defined as “urn:nfc:handover:”.

The message generator is responsible for the uniqueness of the payload identifiers encoded into the ID field of the NDEF record header. While identifiers can be strings of length up to 255 characters, it is RECOMMENDED that short, possibly single character, strings are used. However, the generator SHALL NOT use the tilde character (“~”, hexadecimal 7E) at the first string position and a compliant parser SHALL ignore strings starting with a tilde character.

2.5 Carrier Type Identification

The type of carrier described by an Alternative Carrier Record is indirectly given via the Carrier Data Reference link.

If the referenced NDEF record is a Handover Carrier Record (NFC Forum Global Type “Hc”) as defined in this specification, the carrier type is identified by the Carrier Name structure within the payload of the Handover Carrier Record. The syntax of the Carrier Name structure is equivalent to the NDEF record type. The Handover Carrier Record MUST be used by a Handover Requester when an alternative carrier is proposed without configuration data.

If the referenced NDEF record is not a Handover Carrier Record, then the carrier type is identified with the NDEF record payload type (the triple $\{TNF, TYPE_LENGTH, TYPE\}$), and the payload of the record MUST provide carrier-specific configuration information as needed to connect to the carrier. For brevity, this type of NDEF record is called Carrier Configuration Record, but this is only a conceptual name, and no record named such is defined in this specification.

2.6 Carrier Power State

Handover devices with limited power resources might want to activate (that is, power up) carrier circuits only when they are requested to by a handover activity. For this purpose, a device can indicate the power status for each of the proposed or available alternative carriers within the Alternative Carrier Record. The possible values for the Carrier Power State are defined in Table 2. For brevity, we will refer to them as “active”, “inactive”, “activating”, or “unknown”.

If a carrier is declared “active” and carrier configuration data is provided, the peer device MAY immediately use the configuration data and connect to the carrier.

If the carrier power state is “activating”, the peer device SHOULD expect a delay when trying to connect to the carrier because the circuit was not yet powered when the message was sent out. The exact time needed to activate the carrier is dependent on both technology and implementation and cannot be defined here. It MAY be defined by other organizations.

If a Handover Requester proposes carriers with power state “activating”, it MAY wait for the Handover Selector’s Handover Select Message before actually powering the circuit.

If a Handover Selector does acknowledge carriers with power state “activating”, it SHALL immediately start the activation process after returning the Handover Select Message.

A Handover Selector MAY return one or multiple alternative carriers (with carrier configuration data) that are declared “inactive” if the Handover Requester provided more than one alternative in the Handover Request Message. The Handover Requester SHOULD then request exactly one alternative carrier in a subsequent Handover Request Message. The Handover Selector SHALL respond to this request with the alternative carrier that is declared either “active” or “activating”.

If the Handover Selector is an NFC Forum Tag (Static Handover), it MAY provide a number of “inactive” alternative carriers inside a Handover Select Message stored on the tag. This usually means that the user is expected to manually activate carrier circuits on the device. Nevertheless, a Handover Requester could first try to connect to the carrier(s) because they might have been already activated.

The carrier power state “unknown” is used when the device does not directly support an interface for that carrier and can only be reached via the alternative carrier through a router. However, note that the device MUST be able to provide sufficient carrier configuration data for the peer device to connect to the alternative carrier.

The power state “unknown” MAY be used in both Handover Request and Handover Select messages.

If on-demand activation is used, the implementation should keep the carrier(s) active for a time interval long enough to allow the peer device to connect to the carrier. The value depends on the carrier technology.

2.7 Handover Request Collision

A handover request collision happens if both devices simultaneously send a Handover Request Message after the NFC communication link has been established by the underlying peer-to-peer protocol. Note that if a device intends to send a Handover Request message but receives a Handover Request message from the other device before sending, it SHALL NOT send a Handover Request message but instead take the role of a “Handover Selector device”.

If a device detects that a Handover Request collision has occurred (that is, it has sent a Handover Request Message and then received a Handover Request Message), it SHALL compare the random number within the received Handover Request Message against the random number transmitted with the sent Handover Request Message. The random number is contained within a Collision Resolution Record in each Handover Request Message.

The random number comparison shall be performed on both devices with the following steps:

1. Numerically compare the received random number with the sent random number to determine equality or the greater value of both.

2. If both random numbers are numerically equal, then automatic collision resolution will not be possible and the device **SHOULD** generate and send a new Handover Request Message.
3. If the two random numbers are not equal, then compare the least significant bit of the received random number with the least significant bit of the sent random number.
 - If the two bits have the same value, then the device that sent the numerically greater random number **SHALL** assume the role of a “Handover Selector device”, and the device that sent the numerically lower random number **SHALL** silently ignore the received Handover Request Message.
 - If the two bits have a different value, then the device that sent the numerically lower random number **SHALL** assume the role of a “Handover Selector device”, and the device that sent the numerically greater random number **SHALL** silently ignore the received Handover Request Message.

2.8 Message Transport

In the NFC Forum protocol stack, handover messages **SHALL** be exchanged over an LLCP data link connection between a handover requester and a handover selector application. A handover requester application **SHALL** send handover request messages and accept handover select messages as response. A handover selector application **SHALL** accept handover request messages and send handover select messages in response.

A single handover request or handover response message **SHALL** be transmitted as a contiguous sequence of octets within the information field(s) of consecutively numbered I PDUs defined by [LLCP]. The number of octets in the information field of the last I PDU transmitting a specific handover message **SHALL** be equal to the number of data octets that have not been transmitted with the preceding I PDUs. The receiver of a handover message **SHALL** determine the total length of transmitted data by observing the message begin, message end, and record length NDEF header fields of the top-level records comprising the NDEF message.

If a handover selector application is available on the local device, it **SHALL** be registered with the local service discovery component by the service name “urn:nfc:sn:handover”, such that a remote handover requester application is able to use the SDP protocol to learn the assigned service access point address for the handover service or issue a **CONNECT** PDU with the service name parameter set to “urn:nfc:sn:handover”.

A handover selector application registered by the service name “urn:nfc:sn:handover” **SHALL** silently discard connection-less transport mode PDUs.

2.9 Version Handling

Both Handover Request Message and Handover Select Message carry a version number field that **SHALL** be set equal to the version number of the Connection Handover specification, after which the message content is encoded. The version number is divided into a major and a minor part. A change in the minor version number part indicates backward-compatible changes in the specification that do not affect interoperability. A change in the major version number part implies significant modifications in syntax or semantics, and parsers supporting only prior versions **SHALL NOT** further interpret the data.

If version numbers do not match exactly, the following rules apply:

- If a Handover Selector reads a Handover Request message with a version number that differs only in the minor part, it SHALL reply with a Handover Select Message formatted according to its own version number.
- If a Handover Selector reads a Handover Request Message with a version number that differs in the major part and is higher than its own version number, it SHALL reply with an empty Handover Select Message (one that does not contain any Alternative Carrier records), and the version number field SHALL be set to the highest supported value.
- If a Handover Selector reads a Handover Request message with a version number that differs in the major part and is lower than its own version number, it MAY reply with an empty Handover Select Message indicating only the version number conflict, or it MAY reply with a Handover Select Message formatted according to the major version used by the Handover Requester.

2.10 Security Considerations

This section is meant to inform application developers and users of the security limitations in the Negotiated and Static Handover protocol described in this specification.

The Handover Protocol requires transmission of network access data and credentials (the carrier configuration data) to allow one device to connect to a wireless network provided by another device. Because of the close proximity needed for communication between NFC Devices and Tags, eavesdropping of carrier configuration data is difficult, but not impossible, without recognition by the legitimate owner of the devices. Transmission of carrier configuration data to devices that can be brought to close proximity is deemed legitimate within the scope of this specification.

In case the legitimate owner of the devices has concerns over the confidentiality of the data, an additional security analysis is necessary that takes the system in question and the operating environment into account.

3 NDEF Structure

The ABNF definition of the Handover messages and records is given in normative Appendix A.

3.1 Message Definitions

3.1.1 Handover Request Message

The Handover Request message is used by a Handover Requester device to propose a number of alternative carriers to the Handover Selector device. The message **MUST** start with a Handover Request Record that has the message begin (MB) flag set and **MUST** be followed by a number of NDEF records. The last NDEF record closes the message with the message end (ME) flag set.

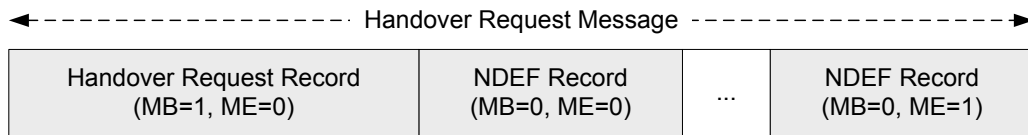


Figure 8: Handover Request Message Structure

Note that a sole Handover Request record with MB and ME set to 1 would not be valid because a Handover Request Message transmits at least one Handover Carrier or Carrier Configuration record to indicate a single alternative carrier.

3.1.2 Handover Select Message

The Handover Select Message is used by the Handover Selector device to acknowledge alternative carriers that were listed in the previously received Handover Request Message. The message **MUST** start with a Handover Select Record that has the message begin (MB) flag set and **MAY** be followed by a number of NDEF records, the last one closing the message with the message end (ME) flag set. In case no additional records are transmitted, the Handover Select Record **SHALL** have both the MB and ME flag set to one.

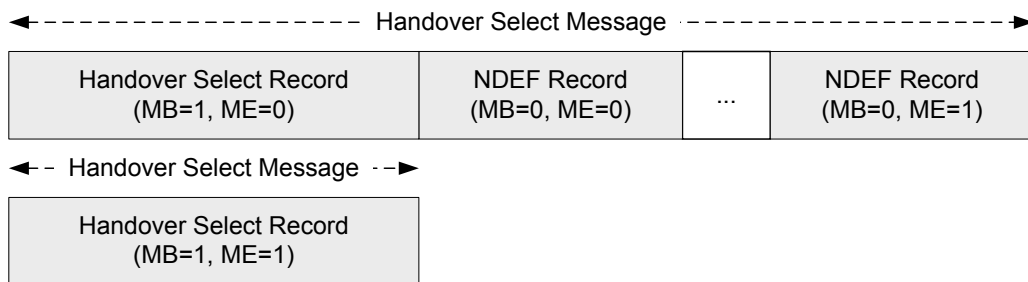


Figure 9: Handover Select Message Structure

Note that a sole Handover Select Record with both MB and ME set to 1 is a valid Handover Select Message that is transmitted if either no matching alternative carrier could be identified or if the version number in the Handover Request Message indicates a message format not supported by the Handover Selector.

3.2 Global Record Definitions

3.2.1 Handover Request Record

The Handover Request Record identifies a list of possible alternative carriers that the Handover Requester device would be able to use for further communication with the Handover Selector. At least a single alternative carrier **MUST** be specified by the Handover Requester. If multiple alternative carriers are specified, the Handover Selector **SHOULD** process the records in order and acknowledge the first appropriate match, if any.

Only Alternative Carrier Records or Collision Resolution Records have a defined meaning in the payload of a Handover Request Record. However, an implementation **SHALL** silently ignore and **SHALL NOT** raise an error if it encounters other unknown record types.

The NFC Forum Well Known Type [NDEF], [NFC RTD] for the Handover Request Record is “Hr” (in NFC binary encoding: 0x48, 0x72).

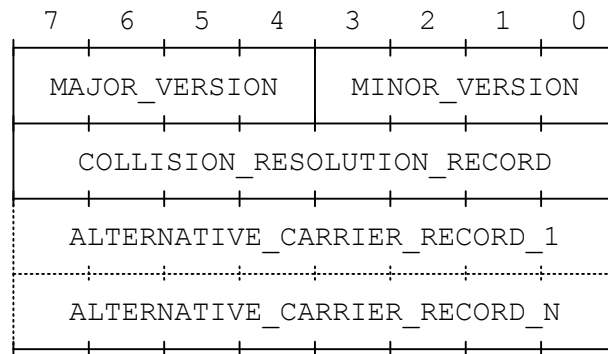


Figure 10: Payload of the Handover Request Record

Semantics for the Handover Request Record:

MAJOR_VERSION: This 4-bit field equals the major version number of the Connection Handover specification and **SHALL** be set to 0x1 by an implementation that conforms to this specification. When an NDEF parser reads a different value, it **SHALL NOT** assume backward compatibility.

MINOR_VERSION: This 4-bit field equals the minor version number of the Connection Handover specification and **SHALL** be set to 0x2 by an implementation that conforms to this specification. When an NDEF parser reads a different value, it **MAY** assume backward compatibility.

COLLISION_RESOLUTION_RECORD: This record contains a 16-bit random number that is used in the collision resolution procedure defined in section 2.7. Only a single Collision Resolution Record **SHALL** be allowed in a Handover Request Message. The Collision Resolution Record is defined in section 3.3.2.

ALTERNATIVE_CARRIER_RECORD: Each record specifies a single alternative carrier that the Handover Requester would be able to use for further communication with the Handover Selector device. The Alternative Carrier Record is defined in section 3.3.1.

3.2.2 Handover Select Record

The Handover Select Record identifies the alternative carriers that the Handover Selector device selected from the list provided within the previous Handover Request Message. The Handover Selector MAY acknowledge zero, one, or more of the proposed alternative carriers at its own discretion.

Only Alternative Carrier Records and Error Records have a defined meaning in the payload of a Handover Select Record. However, an implementation SHALL NOT raise an error if it encounters other record types, but SHOULD silently ignore them.

The NFC Forum Well Known Type [NDEF], [NFC RTD] for the Handover Select record is “Hs” (in NFC binary encoding: 0x48, 0x73).

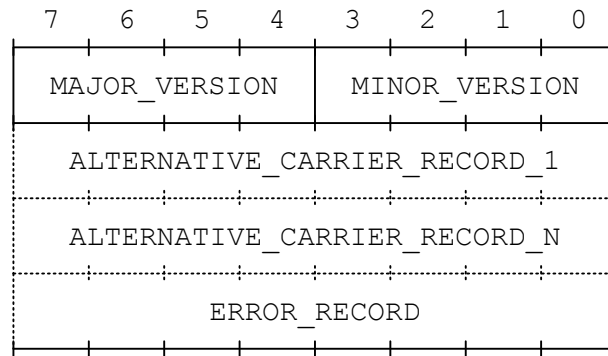


Figure 11: Payload of the Handover Select Record

Semantics for the Handover Select Record:

MAJOR_VERSION: This 4-bit field equals the major version number of the Connection Handover specification and SHALL be set to 0x1 by an implementation that conforms to this specification. When an NDEF parser reads a different value, it SHALL NOT assume backward compatibility.

MINOR_VERSION: This 4-bit field equals the minor version number of the Connection Handover specification and SHALL be set to 0x2 by an implementation that conforms to this specification. When an NDEF parser reads a different value, it MAY assume backward compatibility.

ALTERNATIVE_CARRIER_RECORD: Each record specifies a single alternative carrier that the Handover Selector would be able to use for further communication with the Handover Requester device. The order of the Alternative Carrier Records gives an implicit preference ranking that the Handover Requester SHOULD obey. The Alternative Carrier Record is defined in section 3.3.1.

ERROR_RECORD: This record indicates that the Handover Selector was not able to successfully process the Handover Request Message, either in part or entirely. Only a single Error Record SHALL be allowed in a Handover Select Message. Alternative Carrier Records encoded before an Error Record SHALL be complete and regarded as valid information. Alternative Carrier Records SHALL NOT be encoded after an Error Record. The Error Record is defined in section 3.3.3

3.2.3 Handover Carrier Record

The Handover Carrier Record provides a unique identification of an alternative carrier technology in Handover Request messages when no carrier configuration data is to be provided. If the Handover Selector has the same carrier technology available, it would respond with a Carrier Configuration record with payload type equal to the carrier type (that is, the triples $\{TNF, TYPE_LENGTH, TYPE\}$ and $\{CTF, CARRIER_TYPE_LENGTH, CARRIER_TYPE\}$ match exactly).

The NFC Forum Well Known Type [NDEF], [NFC RTD] for the Handover Carrier Record is “Hc” (in NFC binary encoding: 0x48, 0x63).

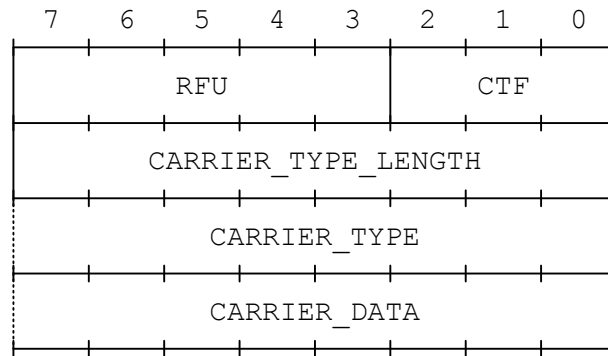


Figure 12: Handover Carrier Record Encoding

Semantics for the Handover Carrier Record:

CTF (Carrier Type Format): This is a 3-bit field that indicates the structure of the value of the *CARRIER_TYPE* field. Allowed values are listed in Table 1; other values SHALL NOT be used. The *CTF* field has the same semantics as the NDEF record *TNF* (Type Name Format) field and further explanation can be found in the NDEF specification [NDEF].

Table 1: Allowed CTF Field Values

Value	Carrier Type Format
0x01	NFC Forum well-known type [NFC RTD]
0x02	Media-type as defined in RFC 2046 [RFC 2046]
0x03	Absolute URI as defined in RFC 3986 [RFC 3986]
0x04	NFC Forum external type [NFC RTD]

CARRIER_TYPE: The value of the *CARRIER_TYPE* field gives a unique identification of the alternative carrier (see section 2.5). The value of the *CARRIER_TYPE* field MUST follow the structure, encoding, and format implied by the value of the *CTF* field.

CARRIER_DATA: A sequence of octets that provide additional information about the alternative carrier enquiry. The syntax and semantics of this data are determined by the *CARRIER_TYPE* field. The number of *CARRIER_DATA* octets is equal to the NDEF record *PAYLOAD_LENGTH* minus the *CARRIER_TYPE_LENGTH* minus 2.

3.3 Local Record Definitions

3.3.1 Alternative Carrier Record

The Alternative Carrier Record is used in the Handover Request Record or the Handover Select Record to describe a single alternative carrier. It SHALL NOT be used elsewhere.

The carrier type structure is the basic identification of an alternative carrier. The possible values are the same as those that can be assigned to the NDEF payload type field (see [NDEF]).

The list of Payload ID References is used to provide additional information by referencing other NDEF records within the Handover Request or Handover Select Message. This list can contain any number of links, including none.

The NFC Forum Well Known Local Type [NDEF], [NFC RTD] for the Alternative Carrier Record is “ac” (in NFC binary encoding: 0x61, 0x63).

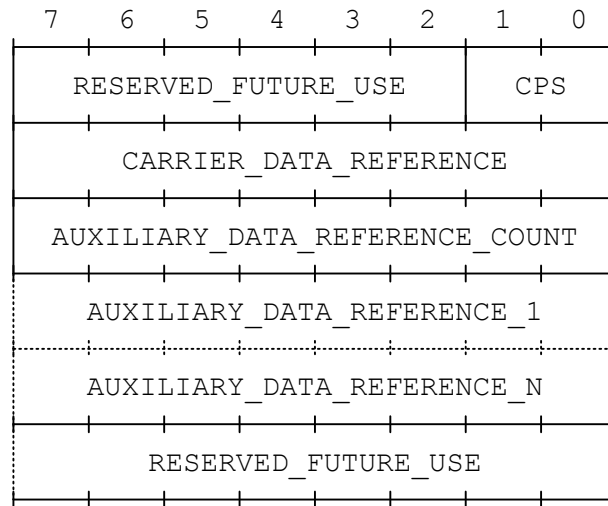


Figure 13: Alternative Carrier Record Layout

Semantics for the Alternative Carrier Record:

CPS (Carrier Power State): This is a 2-bit field that indicates the carrier power state. Possible values are described in Table 2.

Table 2: Carrier Power State Values

Value	Carrier Power State
0x00	Inactive; the carrier is currently off
0x01	Active; the carrier is currently on
0x02	Activating; the device is in the process of activating the carrier, but it is not yet active.
0x03	Unknown; the device is only reachable via the carrier through a router, and it does not directly support an interface for the carrier.

CARRIER_DATA_REFERENCE: The Carrier Data Reference is a pointer to an NDEF record that uniquely identifies the carrier technology. The pointed record MAY be either a Handover Carrier record (see section 3.2.3) or a Carrier Configuration record (see section 2.5). A Carrier Data Reference is encoded as an 8-bit length field that determines the number of the following carrier data reference characters (see Figure 14).

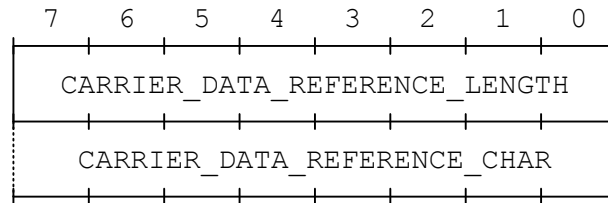


Figure 14: Carrier Data Reference Encoding

AUXILIARY_DATA_REFERENCE_COUNT: This is an 8-bit integer field that defines the number of the following Auxiliary Data References.

AUXILIARY_DATA_REFERENCE: An Auxiliary Data Reference is a pointer to an NDEF record that gives additional information about the alternative carrier. No limitations are imposed on the type of record being pointed to. An Auxiliary Data Reference is encoded as an 8-bit length field that determines the number of the following auxiliary data reference characters.

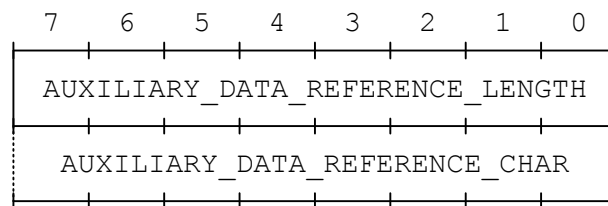


Figure 15: Auxiliary Data Reference Encoding

3.3.2 Collision Resolution Record

The Collision Resolution Record is used in the Handover Request Record to transmit the random number required to resolve a collision of handover request messages. It SHALL NOT be used elsewhere.

The NFC Forum Well Known Local Type [NDEF], [NFC RTD] for the Collision Resolution Record is “cr” (in NFC binary encoding: 0x63, 0x72).

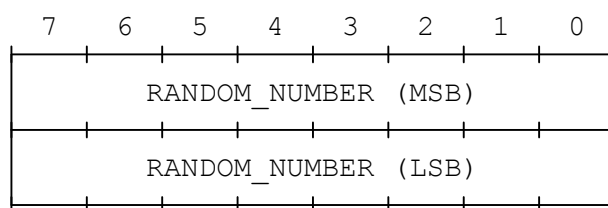


Figure 16: Payload of the Collision Resolution Record

Semantics for the Collision Resolution Record:

RANDOM_NUMBER: This 16-bit field contains an integer number that SHALL be randomly generated before sending a Handover Request Message.

3.3.3 Error Record

The Error Record is used in the Handover Select Record to indicate that the Handover Selector failed to successfully process the most recently received Handover Request Message. It SHALL NOT be used elsewhere.

The NFC Forum Well Known Local Type [NDEF], [NFC RTD] for the Error Record is “err” (in NFC binary encoding: 0x65, 0x72, 0x72).

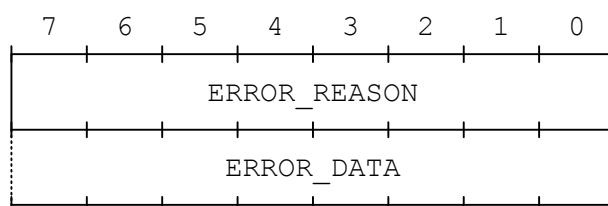


Figure 17: Payload of the Error Record

Semantics for the Error Record:

ERROR_REASON: An 8-bit field that indicates the specific type of error that caused the Handover Selector to return the Error Record. Possible values are described in Table 3.

Table 3: Error Reason Values

Value	Description
0x00	Reserved
0x01	The Handover Request Message could not be processed due to temporary memory constraints. Resending the unmodified Handover Request Message might be successful after a time interval of at least the number of milliseconds expressed in the error data field.
0x02	The Handover Request Message could not be processed due to permanent memory constraints. Resending the unmodified Handover Request Message will always yield the same error condition.

Value	Description
0x03	The Handover Request Message could not be processed due to carrier-specific constraints. Resending the Handover Request Message might not be successful until after a time interval of at least the number of milliseconds expressed in the error data field.
0x04 – 0xFF	Reserved.

ERROR_DATA: A sequence of octets providing additional information about the conditions that caused the handover selector to enter erroneous state. The syntax and semantics of this data are determined by the ERROR_REASON field and are specified in Table 4. The number of octets encoded in the ERROR_DATA field SHALL be determined by the number of octets in the payload of the Error Record minus 1.

Table 4: Contents of the Error Data Field

Error Reason Value	Content of the Error Data field
0x01	An 8-bit unsigned integer that expresses the minimum number of milliseconds after which a Handover Request Message with the same number of octets might be processed successfully. The number of milliseconds SHALL be determined by the time interval between the sending of the error indication and the subsequent receipt of a Handover Request Message by the Handover Selector.
0x02	A 32-bit unsigned integer, encoded with the most significant byte first, that indicates the maximum number of octets of an acceptable Handover Select Message. The number of octets SHALL be determined by the total length of the NDEF message, including all header information.
0x03	An 8-bit unsigned integer that expresses the minimum number of milliseconds after which a Handover Request Message might be processed successfully. The number of milliseconds SHALL be determined by the time interval between the sending of the error indication and the subsequent receipt of a Handover Request Message by the Handover Selector.

A. ABNF Definition of Handover Messages and Records

This section defines the normative requirements for the handover messages and records. The language used is the ABNF format as defined in Augmented BNF for Syntax Specifications: ABNF [RFC 5234].

```

handover_request_message      = handover_request_record 1*NDEF_record
handover_select_message      = handover_select_record *NDEF_record

handover_request_record      = NDEF_header "Hr" handover_request_payload
handover_select_record       = NDEF_header "Hs" handover_select_payload

NDEF_header                  = header_flags record_type_length payload_length
header_flags                  = OCTET
record_type_length            = OCTET
payload_length                = OCTET / UINT32

handover_request_payload     = version_number
                             = collision_resolution_record
                             1*alternative_carrier_record

handover_select_payload      = version_number
                             *alternative_carrier_record
                             error_record

version_number                = OCTET

collision_resolution_record   = random_number

random_number                 = UINT16

alternative_carrier_record    = carrier_state_flags
                             carrier_data_reference
                             auxiliary_data_reference_count
                             *auxiliary_data_reference
                             *reserved_bytes

carrier_state_flags           = OCTET
carrier_data_reference        = payload_reference

auxiliary_data_reference_count = OCTET
auxiliary_data_reference      = payload_reference

payload_reference             = payload_reference_length
                             payload_reference_name

payload_reference_length      = payload_id_length
payload_reference_name        = payload_id

handover_carrier_record       = NDEF_header ID_length "Hc" payload_ID
                             handover_carrier_payload

handover_carrier_payload      = carrier_type_flags carrier_type_length
                             carrier_type [carrier_data]

carrier_type_flags            = OCTET
carrier_type_length           = OCTET
carrier_type                   = 1*OCTET
carrier_data                   = 1*OCTET

payload_id_length             = OCTET
payload_id                    = 1 (ALPHA / DIGIT / "-" / "_" / "." / ":")
                             *(ALPHA / DIGIT / "-" / "_" / "." / ":" / "~")

error_record                  = error_reason *error_data
error_reason                  = OCTET
error_data                    = *OCTET

reserved_bytes                = OCTET

```

Notes:

- *_length and *_count fields. All elements in the grammar with a suffix of “_length” or “_count” contain unsigned 8-bit binary integers indicating the size or number of corresponding fields in the record. For example, payload_reference_count indicates the number of payload_reference fields that follow.
- version_number. This octet consists of two 4-bit subfields. The high-order 4 bits contain the major version number. The low-order 4 bits contain the minor version number. By convention, backward compatibility is assured unless the major version number changes. For example, 0x10 corresponds to version 1.0.
- carrier_state_flags. The two low-order bits of carrier_state_flags reflect the status of the alternative carrier interface with values defined in Table 2. The high-order six bits of carrier_state_flags are reserved for future use and MUST be ignored by version 1.0 implementations.
- carrier_type_flags. The three low-order bits of carrier_type_flags define the format of the carrier_type field with values as defined in Table 1. The high-order five bits of carrier_type_flags are reserved for future use and MUST be ignored by version 1.0 implementations.
- reserved_bytes. This is a placeholder for future backward-compatible extensions of the alternative_carrier_record. If reserved_bytes octets are present, Version 1.0 implementations MUST ignore them.
- NDEF_header (header_flags, record_type_length, payload_length). These fields are as documented in the NFC Data Exchange Format [NDEF] specification.
- UINT16. This field corresponds to a 16-bit unsigned integer in network byte order.
- UINT32. This field corresponds to a 32-bit unsigned integer in network byte order.

B. Revision History

The following table outlines the revision history of Connection Handover.

Table 5: Revision History

Document Name	Revision and Release Date	Status	Change Notice	Supersedes
Connection Handover Candidate Technical Specification	1.0, April 2008	Candidate	None	
Connection Handover Technical Specification	1.1, November 2008	Final		1.0
Connection Handover Technical Specification	1.2, July 2010	Final	Defines the transport of connection handover messages over LLCp. Defines handover request collision resolution.	1.1



NFC Data Exchange Format (NDEF)

Technical Specification

NFC ForumTM

NDEF 1.0

NFCForum-TS-NDEF_1.0

2006-07-24

RESTRICTIONS ON USE

This specification is copyright © 2005-2006 by the NFC Forum, and was made available pursuant to a license agreement entered into between the recipient (Licensee) and NFC Forum, Inc. (Licensor) and may be used only by Licensee, and in compliance with the terms of that license agreement (License). If you are not the Licensee, you are not authorized to make any use of this specification. However, you may obtain a copy at the following page of Licensor's Website: http://www.nfc-forum.org/resources/spec_license after entering into and agreeing to such license terms as Licensor is then requiring. On the date that this specification was downloaded by Licensee, those terms were as follows:

1. LICENSE GRANT.

Licensor hereby grants Licensee the right, without charge, to copy (for internal purposes only) and share the Specification with Licensee's members, employees and consultants (as appropriate). This license grant does not include the right to sublicense, modify or create derivative works based upon the Specification.

2. NO WARRANTIES.

THE SPECIFICATION IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY, COMPLETENESS AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL LICENSOR, ITS MEMBERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE SPECIFICATION.

3. THIRD PARTY RIGHTS.

Without limiting the generality of Section 2 above, LICENSOR ASSUMES NO RESPONSIBILITY TO COMPILE, CONFIRM, UPDATE OR MAKE PUBLIC ANY THIRD PARTY ASSERTIONS OF PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS THAT MIGHT NOW OR IN THE FUTURE BE INFRINGED BY AN IMPLEMENTATION OF THE SPECIFICATION IN ITS CURRENT, OR IN ANY FUTURE FORM. IF ANY SUCH RIGHTS ARE DESCRIBED ON THE SPECIFICATION, LICENSOR TAKES NO POSITION AS TO THE VALIDITY OR INVALIDITY OF SUCH ASSERTIONS, OR THAT ALL SUCH ASSERTIONS THAT HAVE OR MAY BE MADE ARE SO LISTED.

4. TERMINATION OF LICENSE.

In the event of a breach of this Agreement by Licensee or any of its employees or members, Licensor shall give Licensee written notice and an opportunity to cure. If the breach is not cured within thirty (30) days after written notice, or if the breach is of a nature that cannot be cured, then Licensor may immediately or thereafter terminate the licenses granted in this Agreement.

5. MISCELLANEOUS.

All notices required under this Agreement shall be in writing, and shall be deemed effective five days from deposit in the mails. Notices and correspondence to the NFC Forum address as it appears below. This Agreement shall be construed and interpreted under the internal laws of the United States and the Commonwealth of Massachusetts, without giving effect to its principles of conflict of law.

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

Contents

1 Overview	1
1.1 Objectives	1
1.1.1 Design Goals	2
1.1.2 Anti-Goals	2
1.2 References	2
1.3 Administration	4
1.4 Special Word Usage	4
1.5 Name and Logo Usage	4
1.6 Intellectual Property	4
1.7 Glossary	5
2 NDEF Mechanisms	7
2.1 Introduction	7
2.2 Intended Usage	7
2.3 NDEF Encapsulation Constructs	8
2.3.1 Message	8
2.3.2 Record	8
2.3.3 Record Chunks	8
2.4 NDEF Payload Description	9
2.4.1 Payload Length	9
2.4.2 Payload Type	9
2.4.3 Payload Identification	10
2.5 NDEF Mechanisms Test Requirements	11
3 The NDEF Specification	13
3.1 Data Transmission Order	13
3.2 Record Layout	14
3.2.1 MB (Message Begin)	14
3.2.2 ME (Message End)	14
3.2.3 CF (Chunk Flag)	14
3.2.4 SR (Short Record)	15
3.2.5 IL (ID_LENGTH field is present)	15
3.2.6 TNF (Type Name Format)	16
3.2.7 TYPE_LENGTH	17
3.2.8 ID_LENGTH	17
3.2.9 PAYLOAD_LENGTH	17
3.2.10 TYPE	17
3.2.11 ID	17
3.2.12 PAYLOAD	18
3.3 THE NDEF Specification Test Requirements	18
4 Special Considerations	19
4.1 Internationalization	19
4.2 Security	19
4.3 Maximum Field Sizes	19
4.4 Use of URIs in NDEF	19
4.5 Special Consideration Test Requirements	20
A. Revision History	21

Figures

Figure 1. Example of an NDEF Message with a Set of Records.....	8
Figure 2. NDEF Octet Ordering	13
Figure 3. NDEF Record Layout	14
Figure 4. NDEF Short-Record Layout (SR=1).....	15

Tables

Table 1. TNF Field Values	16
Table 2. Revision History	21

Test Requirements

Test Requirements 1. NDEF Mechanisms Test Requirements	11
Test Requirements 2. The NDEF Specification Test Requirements	18
Test Requirements 3. Special Consideration Test Requirements	20

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 <i>MB</i> (Message Begin) flag set.
The last record in the message is marked with the <i>ME</i> (Message End) flag set.
NDEF messages MUST NOT overlap; that is, the <i>MB</i> and the <i>ME</i> 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 <i>CF</i> (Chunk Flag) flag set.
The type of the entire chunked payload MUST be indicated in the <i>TYPE</i> field of the initial record chunk.
The <i>PAYLOAD_LENGTH</i> field of the initial record indicates the size of the data carried in the <i>PAYLOAD</i> field of the initial record only, not the entire payload size.
Each middle record chunk is an NDEF record with the <i>CF</i> flag set.
For each middle record chunk the value of the <i>TYPE_LENGTH</i> and the <i>IL</i> fields MUST be 0.
For each middle record chunk the <i>TNF</i> (Type Name Format) field value MUST be 0x06 (<i>Unchanged</i>).
For each middle record chunk, the <i>PAYLOAD_LENGTH</i> field indicates the size of the data carried in the <i>PAYLOAD</i> field of this single record only.
The terminating record chunk is an NDEF record with the <i>CF</i> flag cleared.
For the terminating record chunk, the value of the <i>TYPE_LENGTH</i> and the <i>IL</i> fields MUST be 0.
For the terminating record chunk, the <i>TNF</i> (Type Name Format) field value MUST be 0x06 (<i>Unchanged</i>).
For the terminating record chunk, the <i>PAYLOAD_LENGTH</i> field indicates the size of the data carried in the <i>PAYLOAD</i> 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 <i>ME</i> (Message End) flag set.
A middle record chunk MUST NOT have the <i>ME</i> (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.

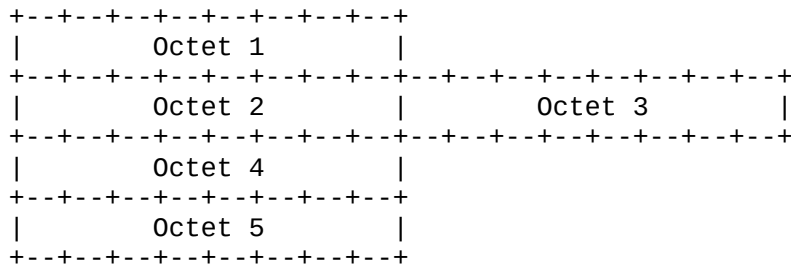


Figure 2. NDEF Octet Ordering

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.

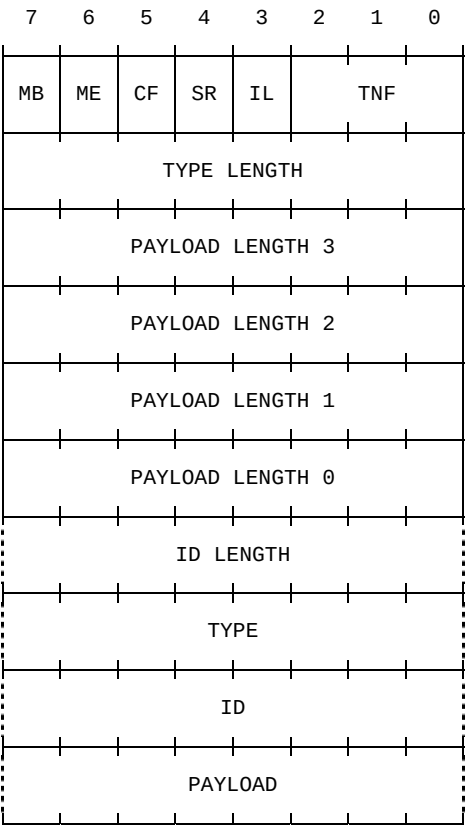


Figure 3. NDEF Record Layout

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.

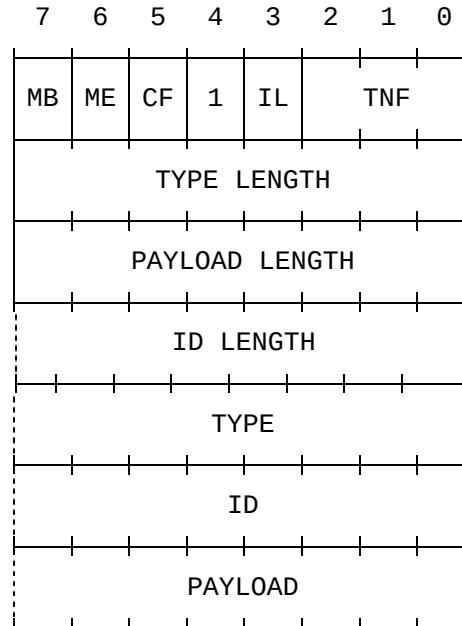


Figure 4. NDEF Short-Record Layout (SR=1)

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	



Type 1 Tag Operation Specification

Technical Specification

NFC Forum™

NFCForum-TS-Type-1-Tag_1.0

2007-07-09

RESTRICTIONS ON USE

This specification is copyright © 2005-2007 by the NFC Forum, and was made available pursuant to a license agreement entered into between the recipient (Licensee) and NFC Forum, Inc. (Licensor) and may be used only by Licensee, and in compliance with the terms of that license agreement (License). If you are not the Licensee, you are not authorized to make any use of this specification. However, you may obtain a copy at the following page of Licensor's Website: http://www.nfc-forum.org/resources/spec_license after entering into and agreeing to such license terms as Licensor is then requiring. On the date that this specification was downloaded by Licensee, those terms were as follows:

1. LICENSE GRANT.

Licensor hereby grants Licensee the right, without charge, to copy (for internal purposes only) and share the Specification with Licensee's members, employees and consultants (as appropriate). This license grant does not include the right to sublicense, modify or create derivative works based upon the Specification.

2. NO WARRANTIES.

THE SPECIFICATION IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY, COMPLETENESS AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL LICENSOR, ITS MEMBERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE SPECIFICATION.

3. THIRD PARTY RIGHTS.

Without limiting the generality of Section 2 above, LICENSOR ASSUMES NO RESPONSIBILITY TO COMPILE, CONFIRM, UPDATE OR MAKE PUBLIC ANY THIRD PARTY ASSERTIONS OF PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS THAT MIGHT NOW OR IN THE FUTURE BE INFRINGED BY AN IMPLEMENTATION OF THE SPECIFICATION IN ITS CURRENT, OR IN ANY FUTURE FORM. IF ANY SUCH RIGHTS ARE DESCRIBED ON THE SPECIFICATION, LICENSOR TAKES NO POSITION AS TO THE VALIDITY OR INVALIDITY OF SUCH ASSERTIONS, OR THAT ALL SUCH ASSERTIONS THAT HAVE OR MAY BE MADE ARE SO LISTED.

4. TERMINATION OF LICENSE.

In the event of a breach of this Agreement by Licensee or any of its employees or members, Licensor shall give Licensee written notice and an opportunity to cure. If the breach is not cured within thirty (30) days after written notice, or if the breach is of a nature that cannot be cured, then Licensor may immediately or thereafter terminate the licenses granted in this Agreement.

5. MISCELLANEOUS.

All notices required under this Agreement shall be in writing, and shall be deemed effective five days from deposit in the mails. Notices and correspondence to the NFC Forum address as it appears below. This Agreement shall be construed and interpreted under the internal laws of the United States and the Commonwealth of Massachusetts, without giving effect to its principles of conflict of law.

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

Contents

1	Introduction.....	1
1.1	Objectives.....	1
1.2	Purpose.....	1
1.3	Applicable Documents or References.....	1
1.4	Administration.....	2
1.5	Special Word Usage.....	2
1.6	Name and Logo Usage.....	2
1.7	Intellectual Property.....	2
1.8	Glossary.....	3
1.9	Convention and notations.....	5
1.9.1	Representation of numbers.....	5
2	Memory Structure and Management.....	6
2.1	General.....	6
2.2	Static Memory Structure.....	6
2.2.1	Memory Map.....	6
2.2.2	Header ROM Format.....	7
2.2.3	UID Format.....	7
2.2.4	Main Read/Write Memory Format.....	7
2.2.5	Block Dh.....	7
2.2.6	Lock Control/Status Bytes.....	7
2.2.7	OTP Bytes.....	8
2.3	Dynamic Memory Structure.....	8
2.3.1	Dynamic Memory Map.....	8
2.3.2	Dynamic Memory Reserved Bytes.....	10
2.3.3	Dynamic Memory Lock Bytes.....	10
2.3.4	Dynamic Memory Area.....	10
2.4	TLV Blocks.....	10
2.4.1	Format.....	10
2.4.2	Location.....	11
2.4.3	Lock Control TLV.....	12
2.4.4	Reserved Memory Control TLV.....	13
2.4.5	NDEF Message TLV.....	13
2.4.6	Proprietary TLV.....	14
2.4.7	NULL TLV.....	14
2.4.8	Terminator TLV.....	14
3	RF Interface.....	15
4	Framing and Transmission Handling.....	16
4.1	Frame Formats.....	16
4.2	Transmission Handling.....	16
5	Command Set.....	17
5.1	State Diagram.....	17
5.2	Tag Command and Response Set.....	17
5.2.1	Static Memory Model.....	17
5.2.2	Dynamic Memory Model.....	17
5.3	Command Format.....	18
5.3.1	Command List.....	18

5.3.2	Command-Response Format	19
5.3.3	Address Operand	19
5.3.4	CRC	20
5.3.5	UID Echo	20
5.4	Command Details	20
5.4.1	Detailed Timing	20
5.4.2	Timing Definitions	20
5.5	REQA and WUPA	21
5.6	Read Identification (RID)	21
5.7	Read All Blocks 0-Eh (RALL)	22
5.8	Read Byte (READ)	22
5.9	Write-Erase Byte (WRITE-E)	24
5.10	Write-No-Erase Byte (WRITE-NE)	25
5.11	Locking	26
5.12	Read Segment (RSEG)	26
5.13	Read 8 Bytes (READ8)	26
5.14	Write-Erase 8 Bytes (WRITE-E8)	27
5.15	Write-No-Erase 8 Bytes (WRITE-NE8)	27
6	NDEF Detection and NDEF Access	28
6.1	NDEF Management	28
6.1.1	Identification as NFC Forum Type 1 Tag	28
6.1.2	Write Permission	28
6.1.3	Confirmation of Presence of NDEF Message in Type 1 Tag	28
6.1.4	Capability Container	28
6.2	Version Treatment	29
6.3	NDEF Storage	31
6.4	Life Cycle	32
6.4.1	General	32
6.4.2	Overview of Life-Cycle States	32
6.4.3	INITIALISED State	32
6.4.4	READ/WRITE State	32
6.4.5	READ ONLY State	33
6.4.6	Determination of Life Cycle State	33
6.5	Rules for Life Cycle Operation	34
6.5.1	Detect NDEF on tag	34
6.5.2	Read NDEF Message	34

Figures

Figure 1: Static Memory Map of the base NFC Forum Type 1 Tag	7
Figure 2: Lock Control/Status Bytes	8
Figure 3: Example Dynamic Memory Map of NFC Forum Type 1 Tag.....	9
Figure 4: Length Field Formats	11
Figure 5: RALL Command/Response Diagram	22
Figure 6: READ Command/Response Diagram.....	22
Figure 7: WRITE-E Command/Response Diagram	24
Figure 8: WRITE-NE Command/Response Diagram	25
Figure 9: Location Of NDEF Message.....	31
Figure 10: Memory Map of Example Smartposter NDEF Message	37
Figure 11: Example Dynamic Memory Map.....	39

Tables

Table 1: Defined TLV blocks.....	11
Table 2: Command-Response Byte Count (Static Memory Model)	17
Table 3: Command-Response Summary (Static Memory Model)	17
Table 4: Command-Response Byte Count (Dynamic Memory Model).....	18
Table 5: Command-Response Summary (Dynamic Memory Model).....	18
Table 6: List of Commands (Static Memory Model)	18
Table 7: List of Additional Commands (Dynamic Memory Model).....	19
Table 8: Format of Address Operand ADD (Static Memory Structure)	19
Table 9: Format of Address Operand ADDS (Dynamic Memory Model).....	19
Table 10: Format of Address Operand ADD8 (Dynamic Memory Model)	20
Table 11: Timing Definitions	21
Table 12: FDT Timing Calculations.....	21
Table 13: Example Coding of the CC Bytes of Block 1	29
Table 14: Rules for Handling of the Version Number	30
Table 15: Example Smartposter NDEF Message.....	36
Table 16: Revision History.....	41

1 Introduction

This specification is part of the NFC Forum documentation about tag types that an NFC Forum device needs to support in reader/writer mode.

This specification documents how an NFC Forum Device SHALL operate an NFC Forum Type 1 tag platform. This is not a specification of the NFC Forum Type 1 tag platform itself.

1.1 Objectives

The purpose of this specification is to document the requirements and to specify, with a set of rules and guidelines, the NFC Forum Device operation and management of the Type 1 tag platform.

This specification assumes that the Collision Detection and Device Activation activities have been performed as documented in the Mode Switch specifications [DIGPROT] & [ANINT] and these have been completed up to the level of making a single Type 1 tag identifier (UID) available.

This specification also defines the data mapping and how the NFC Forum Device detects, reads, and writes NDEF data into the Type 1 tag platform in order to achieve and maintain interchangeability and interoperability.

1.2 Purpose

The purpose of this specification is to document the requirements and to specify, with a set of rules and guidelines, the NFC Forum Device operation and management of a Type 1 Tag.

This specification also defines the data mapping and how the NFC Forum Device detects, reads, and writes NDEF data into the Type 1 tag platform in order to achieve and maintain interchangeability and interoperability.

1.3 Applicable Documents or References

[ISO/IEC 18092]	ISO/IEC 18092, Information Technology- Telecommunications and information exchange between systems- Near Field Communication - Interface and Protocol (NFCIP-1).
[NDEF]	“NFC Data Exchange Format (NDEF)” NFC Forum™, May 2006.
[RFC 2119]	S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Harvard University, March 1997.
[DIGPROT]	“NFC Digital Protocol Specification”, NFC Forum™, to be released
[ANINT]	“NFC Analog Interface Specification”, NFC Forum™, to be released

1.4 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.

1.5 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 Type 1 Tag OperatingType 1 Tag Operation Specification 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.

ATQA

Answer To Request Type A

CC

Capability Container

CRC

Cyclic Redundancy Check

HR0

Header ROM byte 0

lsb

least significant bit

LSB

Least Significant Byte

MMI

Memory Management Information

msb

most significant bit

MSB

Most Significant Byte

NDEF

NFC Data Exchange Format

NFC

Near Field Communications

NMN

NDEF Magic Number

UID

Unique Identifier

URI

Uniform Resource Identifier

URL

Uniform Resource Locator (this is a special case of an URI)

RALL

Read All Command

<i>RAED</i>	Read Command
<i>REQA</i>	Request Command Type A
<i>RFU</i>	Reserved for Future Use
<i>RID</i>	Read ID Command
<i>RTD</i>	Record Type Description
<i>ROM</i>	Read Only Memory
<i>RWA</i>	Read Write Access
<i>SENS_REQ</i>	Sense Request Command
<i>TLV</i>	Tag, Length, Value
<i>TMS</i>	Tag Memory Size
<i>UID</i>	Unique IDentification
<i>VNo</i>	Version Number
<i>WRITE-E</i>	Write with Erase Command
<i>WRITE-NE</i>	Write no Erase Command

1.9 Convention and notations

1.9.1 Representation of numbers

The following conventions and notations apply in this document unless otherwise stated.

- Binary numbers are represented by strings of digits 0 and 1 shown with the most significant bit (msb) left and the least significant bit (lsb) right, “b” is added at the end.
Example: 11110101_b
- Hexadecimal numbers are represented using the numbers 0 - 9 and the characters A – F, an “h” is added at the end. The most significant byte (MSB) is shown on the left, the least significant byte (LSB) on the right.
Example: F5_h
- Decimal numbers are represented as is (without any tailing character).
Example: 245

2 Memory Structure and Management

2.1 General

The NFC Forum Type 1 tag utilises a simple memory model.

There SHALL be two memory model mappings depending on the memory size of the tag:

1. Static memory structure applies for a tag with physical memory size equal to 120 bytes,
2. Dynamic memory model applies for a tag with physical memory size larger than 120 bytes.

The memory SHALL be considered as being divided into blocks containing 8 bytes each.

Each block is numbered from 0 to 15 (E_h) for static memory structure or from 0 to k for dynamic memory structure. The number associated to a block is called the ‘block number’.

The 8 bytes inside each block are numbered from 0 to 7, where byte 0 is the LSB and byte 7 is the MSB of the block.

For the complete tag address space then, byte 0 of block 0 corresponds to ByteAddr = 0 as the LSB.

Byte 7 of block E_h for static memory structure or byte 7 of block k for dynamic memory structure indicates the very MSB.

Unless otherwise stated, within this document the byte ordering when defining packets and messages follows the little-endian byte order.

The next two sections described in details the two memory structures.

2.2 Static Memory Structure

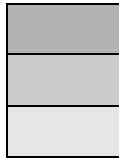
2.2.1 Memory Map

The static memory map of the NFC Forum Type 1 tag, with $HR0 = 11_h$, is shown in Figure 1.

HR0	HR1
11 _h	XX _h

EEPROM Memory Map										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
UID	0	UID-0	UID-1	UID-2	UID-3	UID-4	UID-5	UID-6		Locked
Data	1	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Yes
Data	2	Data8	Data9	Data10	Data11	Data12	Data13	Data14	Data15	Yes
Data	3	Data16	Data17	Data18	Data19	Data20	Data21	Data22	Data23	Yes
Data	4	Data24	Data25	Data26	Data27	Data28	Data29	Data30	Data31	Yes
Data	5	Data32	Data33	Data34	Data35	Data36	Data37	Data38	Data39	Yes
Data	6	Data40	Data41	Data42	Data43	Data44	Data45	Data46	Data47	Yes
Data	7	Data48	Data49	Data50	Data51	Data52	Data53	Data54	Data55	Yes
Data	8	Data56	Data57	Data58	Data59	Data60	Data61	Data62	Data63	Yes
Data	9	Data64	Data65	Data66	Data67	Data68	Data69	Data70	Data71	Yes

EEPROM Memory Map										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
Data	A	Data72	Data73	Data74	Data75	Data76	Data77	Data78	Data79	Yes
Data	B	Data80	Data81	Data82	Data83	Data84	Data85	Data86	Data87	Yes
Data	C	Data88	Data89	Data90	Data91	Data92	Data93	Data94	Data95	Yes
Reserved	D									
Lock/Reserved	E	LOCK-0	LOCK-1	OTP-0	OTP-1	OTP-2	OTP-3	OTP-4	OTP-5	



Reserved for internal use

User Block Lock & Status

OTP bits

Figure 1: Static Memory Map of the base NFC Forum Type 1 Tag.

2.2.2 Header ROM Format

The NFC Forum Type 1 tag includes two bytes of fixed header ROM called HR0 & HR1 as shown in Figure 1. These are not individually addressable by a Read command.

The contents are automatically included in the response packet to certain commands.

HR0 Upper nibble = 0001_b SHALL determine that it is a Type 1, NDEF capable tag.

HR0 Lower nibble = 0001_b SHALL determine static memory map,

≠ 0001_b SHALL determine the dynamic memory map.

HR1 = xx_h is undefined and SHALL be ignored.

2.2.3 UID Format

Block 0 is reserved for the read-only Unique Identification (UID) number.

Byte 7 is reserved for future use.

Byte 6 is the manufacturer's identification code.

Bytes 5, 4, 3, 2, 1, 0 are the unique number.

2.2.4 Main Read/Write Memory Format

The 12 blocks numbered as 1_h to C_h, contain the 96 bytes of general read/write memory.

Each block is individually lockable to become read-only by use of the relevant bits within the lock control bytes, as described in section 2.2.6.

2.2.5 Block Dh

The block numbered as D_h is read-only and reserved for internal use.

2.2.6 Lock Control/Status Bytes

Bytes 0 & 1 of block E_h function as the lock controls for the various memory blocks.

They operate in a bit-wise one-time-programmable fashion.

Figure 2 shows the factory default settings for a Type 1 tag with static memory map.

The individual locking bits can set to 1_b by using a suitable bit mask via a standard write command to the relevant bytes in block number E_h.

This process is irreversible: if one bit of the lock bytes is set to 1_b, it cannot be changed back to 0_b again.

LOCK-0 (Byte 0 of Block E _h)								LOCK-1 (Byte 1 of Block E _h)							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	B5	b4	b3	b2	b1	b0
0 _b = BLOCK-7 Unlocked	0 _b = BLOCK-6 Unlocked	0 _b = BLOCK-5 Unlocked	0 _b = BLOCK-4 Unlocked	0 _b = BLOCK-3 Unlocked	0 _b = BLOCK-2 Unlocked	0 _b = BLOCK-1 Unlocked	1 _b = BLOCK-0 Locked	Not used	1 _b = BLOCK-E Locked	1 _b = BLOCK-D Locked	0 _b = BLOCK-C Unlocked	0 _b = BLOCK-B Unlocked	0 _b = BLOCK-A Unlocked	0 _b = BLOCK-9 Unlocked	0 _b = BLOCK-8 Unlocked

Figure 2: Lock Control/Status Bytes

2.2.7 OTP Bytes

The bytes 2 – 7 of block E_h are allocated as One Time Programmable (OTP) bits and are not defined for NFC Forum purposes.

2.3 Dynamic Memory Structure

2.3.1 Dynamic Memory Map

The NFC Forum Type 1 tag with dynamic memory map is indicated by HR0 = 1y_h where y ≠ 1. In this case a capability container shall be included in the tag memory containing information about the physical memory size, see section 6.1.4.

An example of the dynamic memory map representation of the NFC Forum Type 1 tag with HR0 = 1y_h, where y ≠ 1, is shown in Figure 3.

HR0	HR1
1y _h	xx _h

EEPROM Memory Map										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
UID	0 _h	UID-0	UID-1	UID-2	UID-3	UID-4	UID-5	UID-6		Locked
Data	1 _h	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Yes
Data	2 _h	Data8	Data9	Data10	Data11	Data12	Data13	Data14	Data15	Yes
Data	3 _h	Data16	Data17	Data18	Data19	Data20	Data21	Data22	Data23	Yes
Data	4 _h	Data24	Data25	Data26	Data27	Data28	Data29	Data30	Data31	Yes
Data	5 _h	Data32	Data33	Data34	Data35	Data36	Data37	Data38	Data39	Yes
Data	6 _h	Data40	Data41	Data42	Data43	Data44	Data45	Data46	Data47	Yes

EEPROM Memory Map										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
Data	7 _h	Data48	Data49	Data50	Data51	Data52	Data53	Data54	Data55	Yes
Data	8 _h	Data56	Data57	Data58	Data59	Data60	Data61	Data62	Data63	Yes
Data	9 _h	Data64	Data65	Data66	Data67	Data68	Data69	Data70	Data71	Yes
Data	A _h	Data72	Data73	Data74	Data75	Data76	Data77	Data78	Data79	Yes
Data	B _h	Data80	Data81	Data82	Data83	Data84	Data85	Data86	Data87	Yes
Data	C _h	Data88	Data89	Data90	Data91	Data92	Data93	Data94	Data95	Yes
Reserved	D _h									
Lock/Reserved	E _h	LOCK-0	LOCK-1	OTP-0	OTP-1	OTP-2	OTP-3	OTP-4	OTP-5	
Lock/Reserved	F _h	LOCK-2	LOCK-3							
Data	10 _h	Data96	Data97	Data98	Data99	Data100	Data101	Data102	Data103	Yes
Data	11 _h	Data104	Data105	Data106	Data107	Data108	Data109	Data110	Data111	Yes
Data	12 _h	Data112	Data113	Data114	Data115	Data116	Data117	Data118	Data119	Yes
Data	13 _h	Data120	Data121	Data122	Data123	Data124	Data125	Data126	Data127	Yes
Data	14 _h	Data128	Data129	Data130	Data131	Data132	Data133	Data134	Data135	Yes
Data	15 _h	Data136	Data137	Data138	Data139	Data140	Data141	Data142	Data143	Yes
Data	.	Data144	Data145	Data146	Data147	Data148	Data149	Data150	Data151	Yes
Data	.	Data152	Data153	Data154	Data155	Data156	Data157	Data158	Data159	Yes
Data	.	Data160	Data161	Data162	Data163	Data164	Data165	Data166	Data167	Yes
Data	.	Data168	Data169	Data170	Data171	Data172	Data173	Data174	Data175	Yes
Data	.	Data176	Data177	Data178	Data179	Data180	Data181	Data182	Data183	Yes
Data	.	Data184	Data185	Data186	Data187	Data188	Data189	Data190	Data191	Yes
Lock/Reserved	.	LOCK-x	LOCK-x	LOCK-x	LOCK-x	LOCK-x	LOCK-x	LOCK-x	LOCK-x	
Lock/Reserved	k	LOCK-x	LOCK-x	LOCK-x	LOCK-x	LOCK-x	LOCK-x	LOCK-x	LOCK-x	

Figure 3: Example Dynamic Memory Map of NFC Forum Type 1 Tag.

In Figure 3, each memory block is numbered from 0 to k.

NOTE Dynamic lock bytes and reserved bytes might be located at any byte address in between or at the end of the data area starting from block 0Fh.

Compared to the static memory structure, the dynamic memory structure shall contain configuration information to describe details of dynamic lock bits and to identify reserved memory areas in the data area using the Lock Control TLV and the Memory Control TLV.

The capability container and TLV data areas are not shown in Figure 3, for an example refer to Appendix A.2.

2.3.2 Dynamic Memory Reserved Bytes

These bytes belong to *Reserved* memory areas and SHALL be ignored / jumped over during read and write parsing operations of NFC Forum data. The location of *Reserved* bytes SHALL be identified by one or more Memory Control TLV blocks, as described in section 2.4.

2.3.3 Dynamic Memory Lock Bytes

A tag with an dynamic memory structure contains two kinds of lock bytes:

1. Static lock bytes as specified in section 2.2.6.
2. Dynamic memory lock bytes.

The position of the dynamic memory lock bytes within the tag memory may change.

2.3.4 Dynamic Memory Area

The additional dynamic memory area is located from block F_h onwards.

The available data area for the dynamic memory structure is contained from block 1 up to the last block of the memory including the 96 bytes of the static memory structure and excluding static and dynamic lock bytes and reserved bytes.

Addressing of memory blocks is relative to and includes Block 0.

The available data area capacity in bytes is equal to:

$$8 \cdot (k - 3) - \text{dynamicLockBytes} - \text{dynamicReservedBytes}$$

This calculation includes the data area of the static memory structure equal to 96 bytes and discounts blocks 0, D_h & E_h .

2.4 TLV Blocks

2.4.1 Format

A TLV block consists of one to three fields:

- T** (tag field, or T field) identifies the type of the TLV block and consists of a single byte encoding a number from 00h to FFh. The tag values 04h to FCh and FFh are reserved for future use by the NFC Forum.
- L** (length field, or L field) provides the size in bytes of the value field. It has two different formats composed of one, or three bytes. The NFC Forum device SHALL understand both two length field formats. Figure 4 shows the two different length field structures. However, depending on the tag field value, the length field may not be present.

One byte format: The NFC Forum device SHALL use the one byte format to code the length of the value field between 00h and FEh bytes. The NFC Forum device SHALL interpret this byte as a cardinal if the value is between 00h and FEh. If it contains FFh, the NFC Forum device SHALL interpret the value as flag that specifies that the length field is composed of more than one byte.

Three consecutive bytes format: The NFC Forum device SHALL use this format to code the length of the value field between 00FFh and FFFEh bytes. The first byte is assumed to be a flag

equal to FFh indicating that two more bytes are present. The NFC Forum device SHALL interpret the two more bytes as a word. The NFC Forum device SHALL interpret this word as a cardinal if the value is between 00FFh and FFFEh. The value FFFFh is reserved for future use (RFU).

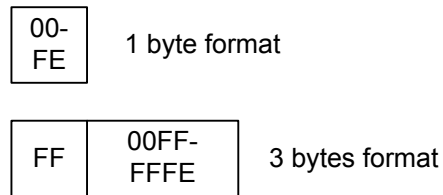


Figure 4: Length Field Formats

V (value field, or V field). If the length field is equal to 00h or there is no length field, the value field is not present, i.e. the TLV block is empty. If there is the length field and it indicates a length N bigger than zero (N>0), the value field consists of N consecutive bytes.

Table 1 lists the TLV blocks defined by this document that are described in the following sections.

Table 1: Defined TLV blocks

TLV block name	Tag Field Value	Short Description
NULL TLV	00 _h	May be used for padding of memory areas and the NFC Forum Device SHALL ignore this
Lock Control TLV	01 _h	Defines details of the lock bytes
Memory Control TLV	02 _h	Identifies reserved memory areas
NDEF Message TLV	03 _h	Contains the NDEF message
Proprietary TLV	FD _h	Tag proprietary information
Terminator TLV	FE _h	Last TLV block in the data area

2.4.2 Location

The NFC Forum device SHALL recognize and interpret the TLV blocks in a specific order inside the data area according to the following rules:

- NDEF Message TLVs and Proprietary TLVs are present after all Lock Control TLVs and Memory Control TLVs.
- If present the Terminator TLV is the last TLV block on the Type 1 tag platform.

NULL TLV and Terminator TLV are the only TLV blocks that are 1 byte long (e.g. composed of only the Tag field, see below).

NFC Forum Devices SHALL ignore and jump over those TLV blocks that make use of reserved Tag field values. To jump over a TLV block with reserved Tag field values, the NFC Forum device SHALL read the length field to understand the length of the value field.

NOTE Future definitions of TLV blocks composed of only the Tag field are not backward compatible with this NFC Forum specification.

2.4.3 Lock Control TLV

The Lock Control TLV can be present inside the Type 1 tag platform. An NFC Forum Device SHALL be able to read and process it. The Lock Control TLV provides control information about the lock areas where the dynamic lock bytes are located.

Each Lock Control TLV indicates a single lock area. More lock areas are indicated using more Lock Control TLV blocks. The encoding of the 3 TLV fields of the Lock Control TLV is as follows:

T is equal to 01h.

L is equal to 03h.

V is composed of 3 bytes that uniquely identify the position and the size of the lock area, and the number of bytes locked by each bit of the dynamic lock bytes. The 3 bytes are encoded in the following way:

- Position, MSB. It codes the position inside the tag memory of the lock area. The position byte consists of 2 parts (to calculate the bytes address from the position byte see below):
- PagesAddr, most significant nibble (4 bits), coded as number of pages (0h=0...Fh=15) and
- ByteOffset, least significant nibble, coded as number of bytes (0h=0...Fh=15).
- Size, middle byte, coded as number of bits (01h=1...FFh=255, 00h=256). It indicates the size in bits of the lock area i.e. the number of dynamic lock bits. If the number of dynamic lock bits is not a multiple of 8, they are stored inside the dynamic lock bytes as explained in the description of the default setting of the dynamic lock bits.
- Page control, LSB. The page control provides general control information: the size in bytes of a page, and the number of bytes that each dynamic lock bit is able to lock. Page control byte is split up into two nibbles of 4 bits each:
 - BytesPerPage: least significant nibble, coded as 2^n (0h=RFU, 1h=1...Fh=15). It indicates the number of bytes per page.
 - BytesLockedPerLockBit: most significant nibble, coded as 2^n (0h=RFU, 1h=1...Fh=15). It indicates the number of bytes that each dynamic lock bit is able to lock.

The NFC Forum device SHALL calculate the byte address (ByteAddr) of the beginning of the lock area in the following way:

$$\text{ByteAddr} = \text{PageAddr} \cdot 2^{\text{BytesPerPage}} + \text{ByteOffset}$$

The ByteAddr is calculated from the beginning of the overall memory of the tag, ie Byte 0 of Block 0 is indicated by ByteAddr equal to 0.

The ByteAddr are used to read and write the relative lock area using the appropriate tag access commands. The page definition has nothing to do with the block definition used by tag access commands.

An example of use of the BytesLockedPerLockBit is as follows. If the memory area locked by a single dynamic lock bit is 8 bytes, then the BytesLockedPerLockBit is equal to 3, i.e. $2^{\text{BytesLockedPerLockBit}} = 2^3 = 8$ bytes.

NOTE The Lock Control TLV might be skipped if a Type 1 tag platform is in READ-ONLY state. Lock Control TLV blocks can be replaced by Reserved Memory

Control TLV indicating the same memory areas for Type 1 tag platform in READ-ONLY state.

2.4.4 Reserved Memory Control TLV

The Reserved Memory Control TLV can be present inside the Type 1 tag platform, and an NFC Forum Device SHALL be able to read and process it. It provides control information about the location and the size of the reserved byte area.

If the vendor delivers the Type 1 tag platform in the READ-ONLY state, the NFC Forum device MAY use the Reserved Memory Control TLV to indicate control information for a mix of reserved and lock areas.

The encoding of the 3 TLV fields of the Reserved Memory Control TLV is:

T is equal to 02h.

L is equal to 03h.

V is composed of 3 bytes that uniquely identify the position and the size of the reserved area. The 3 bytes are encoded in the following way:

- Position, MSB. It codes the position inside the tag of the reserved area. The Position byte consists of 2 parts (to calculate the bytes address from the position byte see below):
- PagesAddr, most significant nibble, coded as number of pages (0h=0...Fh=15) and
- ByteOffset, least significant nibble, coded as number of bytes (0h=0...Fh=15).
- Size, middle byte, coded as number of bytes (1h=1, FFh=255, 0h=256). It indicates the size in bytes of the reserved area.
- Partial Page Control, LSB. The partial page control provides the size in bytes of a page. It is split up into two nibbles of 4 bits each:
 - BytesPerPage nibble: least significant nibble, coded as 2^n (0h=RFU, 1h=1...Fh=15). It indicates the number of bytes per page.
 - Most significant nibble is RFU.

The NFC Forum device SHALL calculate the byte address (ByteAddr) of each reserved area in the following way:

$$ByteAddr = PageAddr \cdot 2^{BytesPerPage} + ByteOffset$$

The ByteAddr is calculated from the beginning of the overall memory of the tag, ie Byte 0 of Block 0 is indicated by ByteAddr equal to 0.

The page definition has nothing to do with the block definition used by tag access commands.

2.4.5 NDEF Message TLV

The NDEF Message TLV is always present inside the Type 1 tag platform. It stores the NDEF message inside the Value field (see [NDEF]). The NFC Forum device SHALL be able to read and process the first (or mandatory) NDEF message. Further NDEF Message TLV blocks can be present.

The encoding of the 3 TLV fields of the NDEF Message TLV is:

T is equal to 03h.

L is equal to the size in bytes of the stored NDEF message.

V stores the NDEF message (see [NDEF]).

An empty NDEF Message TLV is defined as an NDEF Message TLV with L field equal to 00h, and no V field (i.e. no NDEF message is present in the V field, see [NDEF]).

A non-empty NDEF Message TLV can contain either empty or non-empty NDEF messages.

2.4.6 Proprietary TLV

The Proprietary TLV contains proprietary information. A Type 1 tag platform contains zero, one or more Proprietary TLV. The NFC Forum device MAY ignore the data contained in this TLV block.

The encoding of the 3 TLV fields of the Proprietary TLV is:

T is equal to FDh.

L is equal to the size in bytes of the proprietary data in the Value field.

V contains any proprietary data.

2.4.7 NULL TLV

The NULL TLV can be used for padding of the data area. A Type 1 tag platform contains zero, one or more NULL TLV. The NFC Forum device SHALL ignore and jump over this TLV block. NULL TLV is composed of 1 byte tag field.

The encoding of the T field of the NULL TLV is:

T is equal to 00h.

L is not present.

V is not present.

2.4.8 Terminator TLV

The Terminator TLV can be present inside the Type 1 tag platform, and an NFC Forum device SHALL be able to read and process it. The Terminator TLV is the last TLV block in the data area. Terminator TLV is composed of 1 byte tag field.

The encoding of the T field of the Terminator TLV is:

T is equal to FEh.

L is not present.

V is not present.

3 RF Interface

The RF interface of the NFC Forum Device required for operation in NFC Forum Reader/Writer mode is defined in [ANINT].

4 Framing and Transmission Handling

4.1 Frame Formats

The frame formats used by the NFC Forum device to operate with the NFC Forum Type 1 tag are given in [DIGPROT].

4.2 Transmission Handling

The transmission handling of the commands/responses used for initialization, collision detection, device activation activities and selection of the Type 1 tag by the NFC Forum device are defined in [DIGPROT].

Command/responses for operation according to this specification are given in chapter 5.

5 Command Set

5.1 State Diagram

The basic state chart for operation of the NFC Forum Type 1 tag is shown in [DIGPROT].

5.2 Tag Command and Response Set

5.2.1 Static Memory Model

Commands used for the Type 1 tag with the static memory map SHALL generate a response comprised of a number of bytes as shown in Table 2.

Table 2: Command-Response Byte Count (Static Memory Model)

Command	Command bytes	Response bytes
RALL	9	124
READ	9	4
WRITE-E	9	4
WRITE-NE	9	4

Details of the sequence of Command and Response bytes for the operation of the Type 1 tag with the static memory map are shown in Table 3.

Table 3: Command-Response Summary (Static Memory Model)

Command-Response Summary Table																			
Greyed-out frames are dummy frames – their data content SHALL be 00h																			
Command									Response										
RALL	00h	00h	UID 0	UID 1	UID 2	UID 3	CR C1	CR C2	HR0	HR1	UID 0						OTP 5	CR C1	CRC 2
READ	ADD	00h	UID 0	UID 1	UID 2	UID 3	CR C1	CR C2	ADD	DAT	CR C1	CR C2							
WRITE -E	ADD	DAT	UID 0	UID 1	UID 2	UID 3	CR C1	CR C2	ADD	DAT	CR C1	CR C2							
WRITE -NE	ADD	DAT	UID 0	UID 1	UID 2	UID 3	CR C1	CR C2	ADD	DAT	CR C1	CR C2							

A two-byte CRC, as defined in [DIGPROT], SHALL be appended to the end of commands and responses as shown in Table 3.

5.2.2 Dynamic Memory Model

The additional Command-Response bytes required for access to the dynamic memory model are shown in Table 4 and Table 5.

Table 4: Command-Response Byte Count (Dynamic Memory Model)

Command	Command bytes	Response bytes
RSEG	16	131
READ8	16	11
WRITE-E8	16	11
WRITE-NE8	16	11

Table 5: Command-Response Summary (Dynamic Memory Model)

Command															
RSEG	ADD S	00h	00h	00h	00h	00h	00h	00h	00h	UI D0	UI D1	UI D2	UI D3	CRC 1	CRC 2
READ8	ADD 8	00h	00h	00h	00h	00h	00h	00h	00h	UI D0	UI D1	UI D2	UI D3	CRC 1	CRC 2
WRITE-E8	ADD 8	DAT0	DAT1	DAT2	DAT3	DAT4	DAT5	DAT6	DAT7	UI D0	UI D1	UI D2	UI D3	CRC 1	CRC 2
WRITE-NE8	ADD 8	DAT0	DAT1	DAT2	DAT3	DAT4	DAT5	DAT6	DAT7	UI D0	UI D1	UI D2	UI D3	CRC 1	CRC 2

Response													
ADD S	DAT0	DAT1	DAT2	DAT3	DAT4	DAT5	DAT6	DAT7	...	DAT 127	CRC1	CRC2	
ADD8	DAT0	DAT1	DAT2	DAT3	DAT4	DAT5	DAT6	DAT7	CRC1	CRC2			
ADD8	DAT0	DAT1	DAT2	DAT3	DAT4	DAT5	DAT6	DAT7	CRC1	CRC2			
ADD8	DAT0	DAT1	DAT2	DAT3	DAT4	DAT5	DAT6	DAT7	CRC1	CRC2			

5.3 Command Format

5.3.1 Command List

Table 6: List of Commands (Static Memory Model)

Command		Command Code (7-bits)							Comment (all commands are independent)
		msb			lsb				
		b7	b6	b5	b4	b3	b2	b1	
RALL	00 _h	0	0	0	0	0	0	0	Read All (all bytes)
READ	01 _h	0	0	0	0	0	0	1	Read (a single byte)
WRITE-E	53 _h	1	0	1	0	0	1	1	Write-with-erase (a single byte)
WRITE-NE	1A _h	0	0	1	1	0	1	0	Write-no-erase (a single byte)

The Type 1 tag with the static memory model will ignore any other command code bit patterns than those commands shown in Table 6.

Table 7: List of Additional Commands (Dynamic Memory Model)

Command		Command Code (7-bits)							Comment (all commands are independent)
		msb				lsb			
		b7	b6	b5	b4	b3	b2	b1	
RSEG	10 _h	0	0	1	0	0	0	0	Read Segment
READ8	02 _h	0	0	0	0	0	1	0	Read (eight bytes)
WRITE-E8	54 _h	1	0	1	0	1	0	0	Write-with-erase (eight bytes)
WRITE-NE8	1B _h	0	0	1	1	0	1	1	Write-no-erase (eight bytes)

The Type 1 tag with the dynamic memory model will ignore any other command code bit patterns than those commands shown in Table 6 and Table 7.

5.3.2 Command-Response Format

8-bit operand and data frames, as defined in [DIGPROT], SHALL follow all commands listed in Table 6 and Table 7.

5.3.3 Address Operand

The format of the address operand ‘ADD’ for the READ, WRITE-E & WRITE-NE commands of the Type 1 tag with static memory model, SHALL be as shown in Table 8.

Table 8: Format of Address Operand ADD (Static Memory Structure)

Address operand ‘ADD’								
Block = select one of blocks 0 _h – E _h								
Byte = select one of bytes 0 – 7								
msb					lsb			
b8	b7	b6	b5	b4	b3	b2	b1	
0 _b	Static Block				Byte			

The format of the address operand ‘ADDS’ for the RSEG command of the Type 1 tag with the dynamic memory model SHALL be as shown in Table 9.

Table 9: Format of Address Operand ADDS (Dynamic Memory Model)

Address operand ‘ADDS’								
Segment = select one of the Segments 0 _h – F _h								
msb					lsb			
b8	b7	b6	b5	b4	b3	b2	b1	
Segment					0 _b	0 _b	0 _b	0 _b

The format of the block address operand ‘ADD8’ for the READ8, WRITE-E8 & WRITE-NE8 commands of the Type 1 tag with the dynamic memory model SHALL be as shown in Table 10.

Table 10: Format of Address Operand ADD8 (Dynamic Memory Model)

Address operand ‘ADD8’							
Block = select one of the 8-byte blocks 00 _h – FF _h							
msb				lsb			
b8	b7	b6	b5	b4	b3	b2	b1
Global Block							

5.3.4 CRC

The CRC operation SHALL be as defined in [DIGPROT].

5.3.5 UID Echo

The NFC Forum Device in NFC Forum Reader/Writer Mode SHALL execute a single Type 1 tag selection feature as defined in [DIGPROT]. This SHALL result in provision of a single identifier comprised of the lower four bytes of UID.

All subsequent commands used to communicate with this Type 1 tag for operation as described in this specification SHALL include these lower four bytes of UID as part of the proprietary Read and Write commands. If the four lower bytes of UID do not match, then the Type 1 tag will halt operation and remain in ‘READY’ state, as defined in [DIGPROT], waiting for the next valid command.

5.4 Command Details

5.4.1 Detailed Timing

The detailed command timing of a single bit period is defined in [DIGPROT].

5.4.2 Timing Definitions

The timing definitions for commands covered by this document are given in Table 11 below.

Table 11: Timing Definitions

Timing & Description Definitions (used in the command sequence descriptions)		
Name	Description	Specification
RRDD	Reader-Reader Data Delay The time between the end of the last pause of a frame transmitted by the Reader/Writer and the first pause of the next frame to be transmitted by the Reader/Writer.	Minimum: ≥ 28 μs when last bit was 1 ≥ 23 μs when last bit was 0 Maximum: None
DRD	Type1 tag Device Response Delay (Frame Delay Time) “The time between the end of the last pause transmitted by the Reader/Writer and the first modulation edge within the start bit transmitted by the Type 1 tag” (taken from the FDT definition in ISO/IEC 14443-3:2001(E) para 6.1.2)	FDT timing from ISO/IEC 14443-3:2001(E), section 6.1.2 where n: For RALL and READ: n=9 For WRITE_E: n=554 For WRITE_NE: n=281 With tolerance for Digital & Analogue elements of ± 6.5 clock cycles (13.56MHz).
RRD	Reader Response Delay Delay time Type 1 tag to Reader/Writer ie the time between the last modulation transmitted by the Type 1 tag and the first gap transmitted by the Reader/Writer	ISO/IEC 14443-3:2001(E), section 6.1.3 $1172/f_c \cong 86 \mu s$
CE	Command End	
UID-echo	The four least significant UID bytes from block 0 (LSB first)	

Table 12: FDT Timing Calculations

Timing table			
Command	n	$FDT_{bit-1} = 128n + 84$	$FDT_{bit-0} = 128n + 20$
RALL and READ	9	$1236/f_c \approx 91 \mu s$	$1172/f_c \approx 86 \mu s$
WRITE-E	554	$70996/f_c \approx 5236 \mu s$	$70932/f_c \approx 5231 \mu s$
WRITE-NE	281	$36052/f_c \approx 2659 \mu s$	$35988/f_c \approx 2654 \mu s$

NOTE The diagrams in the following sections do not show lead-in, start and end of frame bits.

5.5 REQA and WUPA

These commands are defined in [DIGPROT].

5.6 Read Identification (RID)

This command is defined in [DIGPROT].

5.7 Read All Blocks 0-Eh (RALL)

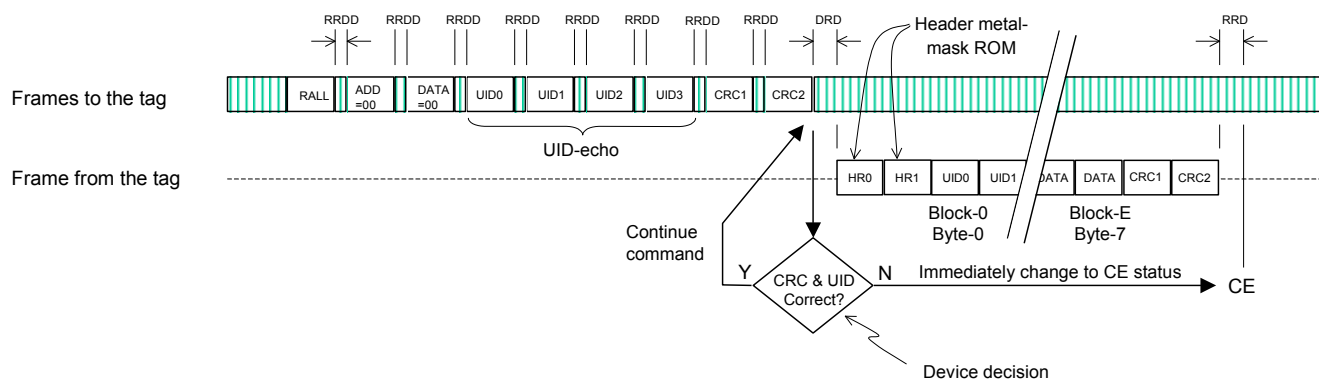


Figure 5: RALL Command/Response Diagram

The RALL command reads-out the two Header ROM bytes and the whole of the static memory blocks 0-E_h.

The Command frame, then Address frame, Data-byte frame, UID-echo frames (with UID data received from previous RID command) & CRC frames SHALL be sent by the NFC Forum Device in NFC Forum Reader/Writer Mode to the tag. However, the Address & Data-bytes SHALL be set to zero.

If the UID and CRC are valid the HR0 & HR1 bytes followed by the contents of memory blocks 0-E_h and the frame CRC bytes will be sent back to the NFC-Forum Device in NFC Forum Reader/Writer Mode.

As a pre-condition this command requires that the tag be in the READY state and afterwards the tag remains in READY state.

5.8 Read Byte (READ)

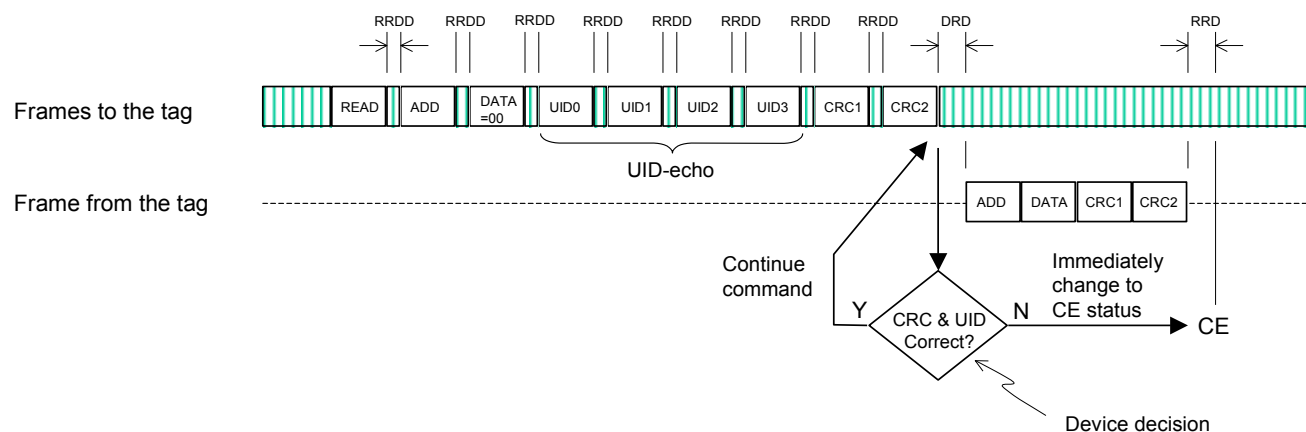


Figure 6: READ Command/Response Diagram

The READ command relates to a single EEPROM memory byte within the static memory model area of blocks 0-E_h. The byte address, (Block number and Byte number), as defined in Table 8, SHALL be sent with the command.

The command frame, then Address frame, Data-byte frame, UID-echo frames (with UID data received from previous RID command) and CRC frames SHALL be sent by the NFC Forum Device in NFC Forum Reader/Writer Mode to the tag. However, the Data-byte SHALL be set to zero.

If the CRC and UID are valid the requested memory data byte is read from memory. The Address, followed by the read data byte and the frame CRC bytes will be sent back to the NFC Forum Device in NFC Forum Reader/Writer Mode.

As a pre-condition this command requires that the tag be in the READY state and afterwards the tag remains in READY state.

5.9 Write-Erase Byte (WRITE-E)

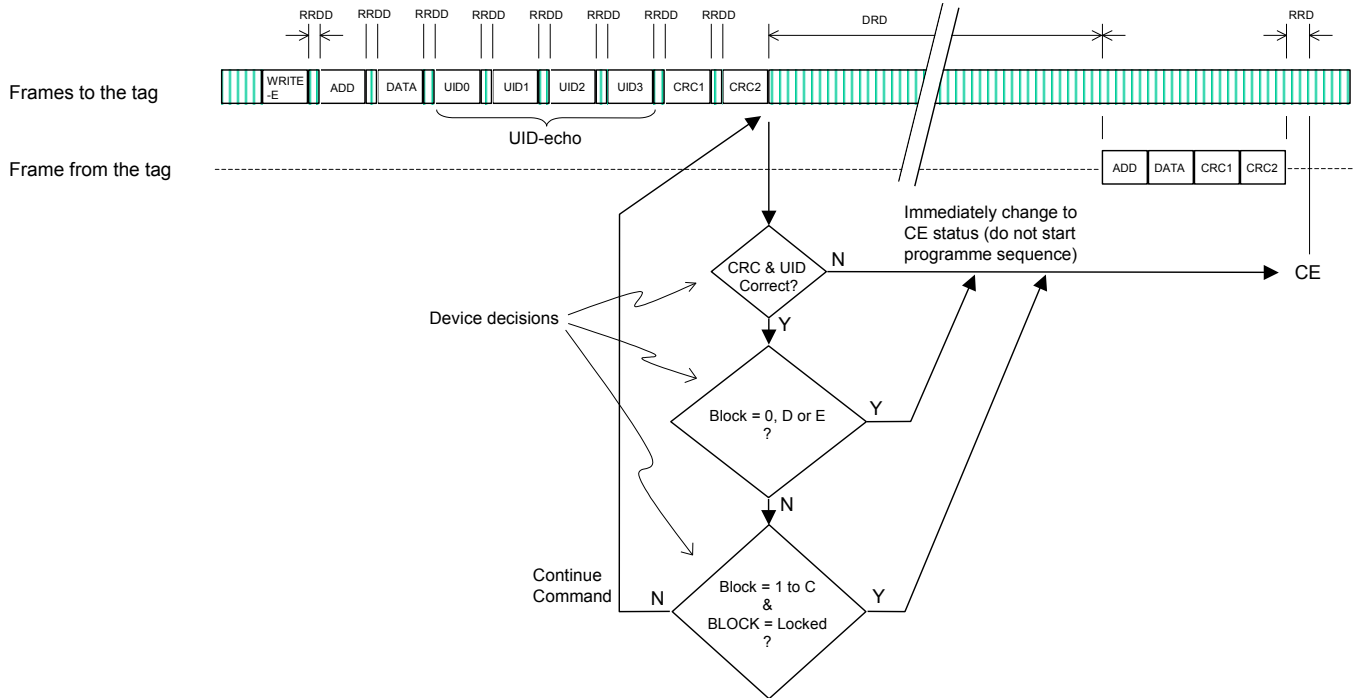


Figure 7: WRITE-E Command/Response Diagram

The WRITE-E (Write-Erase) command relates to an individual memory byte within the static memory model area of blocks 0-E_h. The target byte address, (Block number and Byte number), as defined in Table 8, SHALL be sent with the command. This command performs the ‘normal’ erase-write cycle, (i.e. it erases the target byte before it writes the new data).

If any of BLOCK-0 to BLOCK-D is locked then WRITE-E is barred from those blocks. Additionally, WRITE-E is always barred from Blocks 0, D or E because these are automatically in the locked condition.

The Command frame, then Address frame, Data-byte frame, UID-echo frames (with UID data received from previous RID command) and CRC frames SHALL be sent by the NFC Forum Device in NFC Forum Reader/Writer Mode to the tag.

If the UID and CRC are valid, (and WRITE-E is not barred), the EE memory erase-write cycle is carried out. The byte is then read back from the EE memory. The address, followed by the data byte and the frame CRC bytes are then sent back to the NFC Forum Device in NFC Forum Reader/Writer Mode.

If WRITE-E is barred, the erase-write cycle is skipped – no write operation occurs – and without waiting the programme-time, the tag will enter READY status waiting for a new command.

As a pre-condition this command requires that the tag be in the READY state and afterwards the tag remains in READY state.

5.10 Write-No-Erase Byte (WRITE-NE)

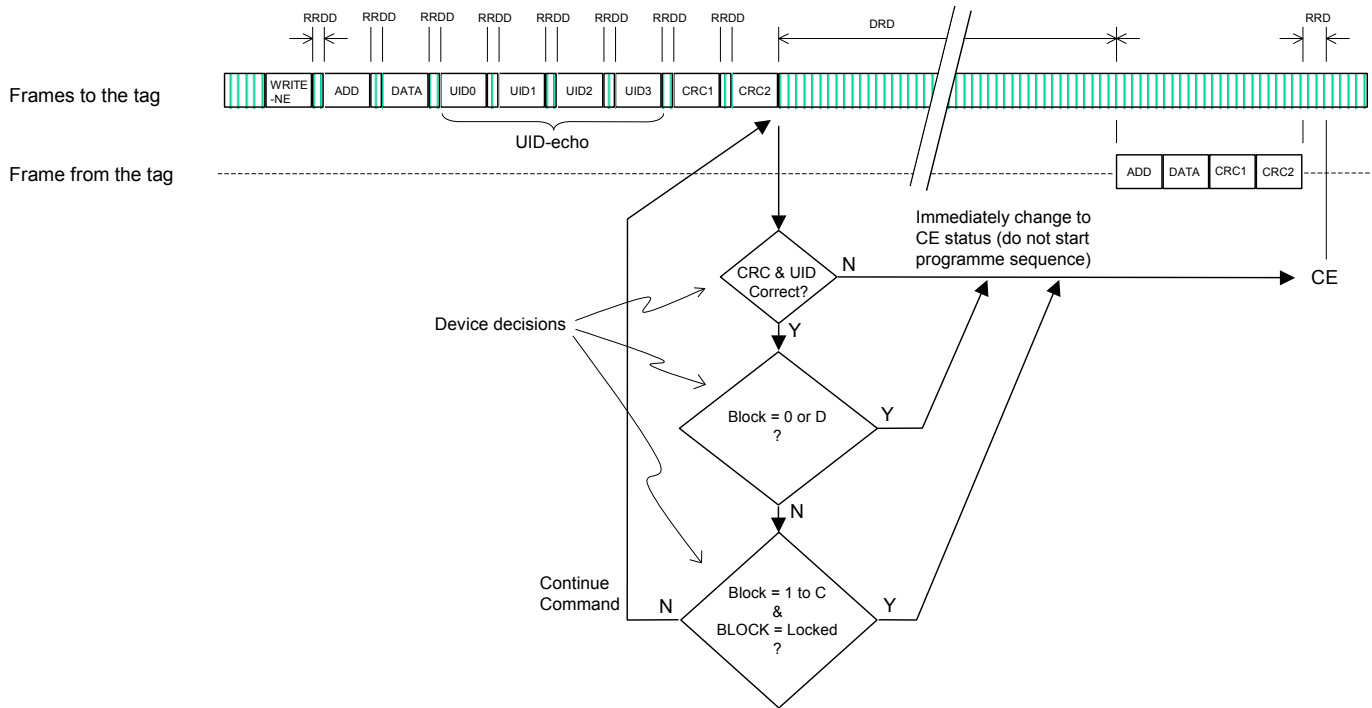


Figure 8: WRITE-NE Command/Response Diagram

The WRITE-NE (Write-no-erase) command relates to an individual memory byte within the static memory model area of blocks 0-E_h. The target byte address, (Block number and Byte number), as defined in Table 8, SHALL be sent with the command. This command does not erase the target byte before writing the new data, and the execution time is approximately half that of the ‘normal’ write command (WRITE-E). Bits can be set but not reset (i.e. data bits previously set to a ‘1’ cannot be reset to a ‘0’).

The WRITE-NE command is available for three main purposes:

- Lock – to set the ‘lock bit’ for a block.
- OTP – to set One-Time-Programmable bits (bytes 2 – 7 of Block-E), where between one and eight OTP bits can be set with a single WRITE-NE command.
- A fast-write in order to reduce overall time to write data to memory blocks for the first time given that the original condition of memory is zero.

If any of BLOCK-1 to BLOCK-C is locked then WRITE-E is barred from that block.

WRITE-NE is not barred from BLOCK-E to allow setting of lock and OTP bits.

The Command frame, then Address frame, Data-byte frame, UID-echo frames (with UID data received from previous RID command) and CRC frames SHALL be sent by the NFC Forum Device in NFC Forum Reader/Writer Mode to the tag.

If the UID and CRC are valid, (and WRITE-NE is not barred), the EE memory write-no-erase cycle is carried out. The byte is then read back from the EE memory. The Address, followed by the Data byte and the frame CRC bytes are then sent back to the NFC Forum Device in NFC Forum Reader/Writer Mode.

If WRITE-NE is barred, the write-no-erase cycle is skipped—no write operation occurs—and without waiting the programme-time, the tag will return to the “READY” state and wait for a new command.

As a pre-condition this command requires that the tag be in the READY state and afterwards the tag remains in READY state.

5.11 Locking

All twelve of the memory blocks 1_h to C_h are separately lockable.

When a block’s ‘lock-bit’ is set to a 1, that block becomes irreversibly frozen as ‘read-only’.

The lock-bits are stored in the Bytes 0 & 1 of BLOCK- E_h .

The WRITE-NE command with appropriate data pattern SHALL be used by the NFC Forum Device in NFC Forum Reader/Writer Mode to set individual lock-bits.

A single WRITE-NE command can be used to set between one and eight lock-bits.

5.12 Read Segment (RSEG)

The RSEG command reads-out a complete segment of memory.

A segment consists of 16 blocks, ie 128 bytes of memory.

The command frames to the Type 1 Tag are similar to the RALL command with the ADD replaced by ADDS, (Address Segment), to select the required segment with the format as defined in Table 9.

The Command-Response summary is given in Table 5.

The Command frame, then Address frame, eight data-byte frames, UID-echo frames (with UID data received from previous RID command) & CRC frames SHALL be sent by the NFC Forum Device in NFC Forum Reader/Writer Mode to the tag. However, the eight data-bytes SHALL be set to zero.

If the UID and CRC are valid then the ADDS, followed by the 128 byte contents of that segment and the frame CRC bytes will be sent back to the NFC-Forum Device in NFC Forum Reader/Writer Mode.

As a pre-condition this command requires that the tag be in the READY state and afterwards the tag remains in READY state.

5.13 Read 8 Bytes (READ8)

The READ8 command reads-out a block of memory.

The command frames to the Type 1 tag are similar to the single byte READ command with the ADD replaced by ADD8, (Address 8), to select the required block with the format as defined in Table 10.

The Command-Response summary is given in Table 5.

The Command frame, then Address frame, eight data-byte frames, UID-echo frames (with UID data received from previous RID command) & CRC frames SHALL be sent by the NFC Forum Device in NFC Forum Reader/Writer Mode to the tag. However, the eight data-bytes SHALL be set to zero.

If the UID and CRC are valid then the ADD8, followed by the 8 data-bytes contents read from that block and the frame CRC bytes will be sent back to the NFC-Forum Device in NFC Forum Reader/Writer Mode.

As a pre-condition this command requires that the tag be in the READY state and afterwards the tag remains in READY state.

5.14 Write-Erase 8 Bytes (WRITE-E8)

The WRITE-E8 command writes with erase to a block of memory.

The command frames to the Type 1 tag are similar to the single byte WRITE-E command with the ADD replaced by ADD8, (Address 8), to select the required block with the format as defined in Table 10.

The Command-Response summary is given in Table 5.

The Command frame, then Address frame, eight data-byte frames for the data to be written, UID-echo frames (with UID data received from previous RID command) & CRC frames SHALL be sent by the NFC Forum Device in NFC Forum Reader/Writer Mode to the tag.

If the UID and CRC are valid then the ADD8, followed by the 8 data-bytes contents just written to that block and the frame CRC bytes will be sent back to the NFC-Forum Device in NFC Forum Reader/Writer Mode.

As a pre-condition this command requires that the tag be in the READY state and afterwards the tag remains in READY state.

5.15 Write-No-Erase 8 Bytes (WRITE-NE8)

The WRITE-E8 command writes with no erase to a block of memory.

The command frames to the Type 1 tag are similar to the single byte WRITE-NE command with the ADD replaced by ADD8, (Address 8), to select the required block with the format as defined in Table 10.

The Command-Response summary is given in Table 5.

The Command frame, then Address frame, eight data-byte frames for the data to be written, UID-echo frames (with UID data received from previous RID command) & CRC frames SHALL be sent by the NFC Forum Device in NFC Forum Reader/Writer Mode to the tag.

If the UID and CRC are valid then the ADD8, followed by the 8 data-bytes contents just written to that block and the frame CRC bytes will be sent back to the NFC-Forum Device in NFC Forum Reader/Writer Mode.

As a pre-condition this command requires that the tag be in the READY state and afterwards the tag remains in READY state.

6 NDEF Detection and NDEF Access

6.1 NDEF Management

6.1.1 Identification as NFC Forum Type 1 Tag

The Type 1 tag has a fixed Header ROM byte called HR0.

To identify the Type 1 tag the high nibble of HR0 SHALL be equal to 0001_b.

When the NFC Forum Device operating in NFC Forum Reader/Writer Mode encounters a tag working to the proprietary protocol as used by the Type 1 tag then it SHALL use this HR0 value to identify whether or not the tag is capable of carrying an NDEF message.

The HR0 value SHALL be made available from the output of the collision detection, device activation and single identifier activities of the Mode Switch as defined in [DIGPROT].

6.1.2 Write Permission

An NFC Forum Device in NFC Forum Reader/Writer Mode SHALL not attempt to write to a tag unless confirmed by HR0 = 1x_h. This pre-qualification SHALL be used to protect accidental writing and corruption of a non-NDEF application tag such as a transit ticket based on an IC operating with the same proprietary protocol but with different HR0 value.

6.1.3 Confirmation of Presence of NDEF Message in Type 1 Tag

Although the qualification of the HR0 value will have identified the tag encountered as a Type 1 tag and hence capable of carrying an NDEF message, there may or may not be an actual NDEF message present.

To further qualify that a valid NDEF message is actually present, a Capability Container (CC) SHALL be used.

The CC SHALL contain NFC Forum management data.

The CC SHALL be assigned to be in the first four bytes of memory block 1.

6.1.4 Capability Container

The CC memory area SHALL not be used to store any application related data.

- Byte 0 when equal to E1_h (NDEF Magic Number) SHALL indicate that NFC Forum defined NDEF Message data is stored in the data area.
- Byte 1 SHALL carry the Version Number (VNo) of this document as supported by the Type 1 tag. The most significant nibble, (the 4 most significant bits), SHALL indicate the major version number, and the least significant nibble, (the 4 least significant bits), SHALL indicate the minor version number. The VNo may change during the life time of an NFC Forum Type 1 tag.
- Byte 2 SHALL indicate the physical tag memory size (TMS) of the Type 1 tag as multipliers of (8 bytes) * (n+1). Examples:
 - 120 bytes are indicated by 0E_h,
 - 256 bytes are indicated by 1F_h,
 - 2048 bytes are indicated by FF_h.

- Byte 3 SHALL indicate the read and write access (RWA) capability of the CC and data area of the Type 1 tag.
 - The most significant nibble (the 4 most significant bits) SHALL indicate the read access condition:
 - The value 0_h indicates read access granted without any security.
 - Any other value is reserved for future use.
 - The least significant nibble (the 4 least significant bits) SHALL indicate the write access condition:
 - The value 0_h indicates write access granted without any security.
 - The value F_h indicates no write access granted at all.
 - Any other value is reserved for future use.

Table 13 shows an example coding of the CC bytes.

This example is related to a Type 1 tag:

- With NFC Forum defined data (byte 0 = E1_h)
- Supporting the version 1.0 (major number 1_h, minor number 0_h) of the mapping document (byte 1 = 10_h),
- With 120 bytes of memory size (byte 2 = 0E_h)
- With read and write access granted without any security (byte 3 = 00_h).

Table 13: Example Coding of the CC Bytes of Block 1

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
NDEF "Magic Number"	Version Number	Tag Memory Size	Read/Write Access	Start of TLV and NDEF Message data area			
NMN	VNo	TMS	RWA	Octet 1	Octet 2	Octet 3	Octet 4
E1 _h	10 _h	0E _h	00 _h	-	-	-	-

6.2 Version Treatment

Byte 1 of the CC contains the Version number, (VNo), of this document as applied to the storage of NDEF Message data within Type 1 tag.

This SHALL be indicated with two numbers: major number version and minor version number.

The rules for the handling of the different document version numbers applied to the Type 1 tag, (called T1VNo), and the one implemented in the NFC Forum device, (called NFCDevVNo), are explained in the cases shown in Table 14.

Table 14: Rules for Handling of the Version Number

No	Version Number Case	Handling
1	Major NFCDevVNo is equal to major T2VNo, and minor NFCDevVNo is bigger than or equal to minor T2VNo	The NFC Forum device SHALL access the Type 1 tag platform and SHALL use all features of the applied mapping document to this Type 1 tag platform.
2	If major NFCDevVNo is equal to major T2VNo, and minor NFCDevVNo is lower than minor T2VNo	Possibly not all features of the Type 1 tag platform can be accessed. The NFC Forum device SHALL use all its features and SHALL access this Type 1 tag platform.
3	If major NFCDevVNo is smaller than major T2VNo	Incompatible data format. The NFC Forum device cannot understand the Type 1 tag platform data. The NFC Forum device SHALL reject this Type 1 tag platform.
4	If major NFCDevVNo is bigger than major T2VNo	The NFC Forum device might implement the support for previous versions of this specification in addition to its main version. In case the NFC Forum device has the support from previous version, it SHALL access the Type 1 tag platform. On the contrary, in case the NFC Forum device has not the support from previous version, it SHALL reject the Type 1 tag platform.

NOTE Future versions of this specification have to define the allowed actions with an NFC Forum Tag with a version number lower than the version number of the NFC Forum device (e.g. whether it is allowed to upgrade the tag to the new version).

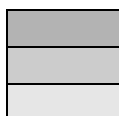
6.3 NDEF Storage

The data format of the NDEF Message is defined in [DIGPROT].

The NDEF Message SHALL be stored inside the value field of the NDEF Message TLV in the data area of the Type 1 tag as shown in Figure 9.

HR0	HR1
11 _h	xx _h

EEPROM Memory Map									
Memory Block	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
0	UID-0	UID-1	UID-2	UID-3	UID-4	UID-5	25 _h		Locked
1	CC0 (NMN) =E1 _h	CC1 (Vno) =10 _h	CC2 (TMS) =0E _h	CC3 (RWA) =00 _h	NDEF Message TLV T=03 _h	NDEF Message TLV L=5A _h	Octet1	Octet2	Yes
2	Octet3	Octet4	Octet5	Octet6	Octet7	Octet8	Octet9	Octet10	Yes
3	Octet11	Octet12	Octet13	Octet14	Octet15	Octet16	Octet17	Octet18	Yes
4	Octet19	Octet20	Octet21	Octet22	Octet23	Octet24	Octet25	Octet26	Yes
5	Octet27	Octet28	Octet29	Octet30	Octet31	Octet32	Octet33	Octet34	Yes
6	Octet35	Octet36	Octet37	Octet38	Octet39	Octet40	Octet41	Octet42	Yes
7	Octet43	Octet44	Octet45	Octet46	Octet47	Octet48	Octet49	Octet50	Yes
8	Octet51	Octet52	Octet53	Octet54	Octet55	Octet56	Octet57	Octet58	Yes
9	Octet59	Octet60	Octet61	Octet62	Octet63	Octet64	Octet65	Octet66	Yes
A	Octet67	Octet68	Octet69	Octet70	Octet71	Octet72	Octet73	Octet74	Yes
B	Octet75	Octet76	Octet77	Octet78	Octet79	Octet80	Octet81	Octet82	Yes
C	Octet83	Octet84	Octet85	Octet86	Octet87	Octet88	Octet89	Octet90	Yes
D									Locked
E	LOCK-0	LOCK-1	OTP-0	OTP-1	OTP-2	OTP-3	OTP-4	OTP-5	OTP



Reserved for internal use

User Block Lock & Status

OTP bits

Figure 9: Location Of NDEF Message

The TLV and NDEF Message storage SHALL start from byte 4 of memory block 1 onwards, up to the maximum capacity of the memory.

6.4 Life Cycle

6.4.1 General

An NFC Forum Type 1 tag can be classified to exist in several states. The state is reflected by the contents of the tag as perceived by the NFC Forum Device in NFC Forum Reader/Writer Mode.

Each state SHALL have its own set of valid operations depending on the current context of the NFC Forum Device. By context, it is meant whether the application is expecting to read from a tag or expecting to write NDEF message onto a tag.

The following sections specify the life-cycle relevant to the NFC Forum Type 1 tag.

6.4.2 Overview of Life-Cycle States

The NFC Forum Device in NFC Forum Reader/Writer Mode SHALL interpret the NFC Forum Type 1 tag to be in one of the following states: INITIALISED, READ/WRITE, and READ-ONLY.

The state SHALL be reflected by the content of the tag.

The state transitions are only relevant for NFC Forum Devices, which are capable of writing Type 1 tags.

The states represented in this section are not related to tag command states as shown in [DIGPROT].

6.4.3 INITIALISED State

A Type 1 tag SHALL be considered to be in INITIALISED state when not in the READ/WRITE or READ ONLY states.

The Capability container as defined in section 6.1.4 shall be present in the Initialised state.

In the case of tags with memory size >120 Bytes, (ie the Dynamic Memory structure), then the Lock Control and Memory Control TLV blocks as defined in section 2.4 shall also be present in the Initialised state. See example in Appendix A.2.

6.4.4 READ/WRITE State

The tag is considered to already contain a valid NDEF message content.

It is available for read and re-write access.

This state SHALL provide the ability to read the NDEF message and also to modify, ie completely over-write the existing NDEF message with a new NDEF message.

This state SHALL be reached via the INITIALISED state.

A Type 1 tag SHALL be detected in READ/WRITE state when:

- CC has byte 0 equal to E1_h, and
- CC has byte 1 with value according to version handling rules of section 6.2, and
- CC has byte 3 equal to 00h (read/write access granted), and
- The data area contains an NDEF Message TLV, and
- The length field of the NDEF Message TLV is different from zero and equal to the actual length of the NDEF message in the value field.

6.4.5 READ ONLY State

This state SHALL be reached via the READ/WRITE or INITIALISED state. In this configuration the CC and the whole data area SHALL be set to read-only. The Type 1 tag SHALL stay in READ-ONLY state for the remaining life cycle.

The tag is considered to contain a valid NDEF message and be available for read only access. It cannot be deleted or overwritten with a new NDEF message.

A Type 1 tag SHALL be detected in READ-ONLY state when:

- CC has byte 0 equal to E1_h, and
- CC has byte 1 with value according to version handling rules of section 6.2, and
- CC area has byte 3 equal to 0F_h, (only read access granted), and
- The data area contains an NDEF Message TLV, and
- The length field of the NDEF Message TLV SHALL be different from zero and equal to the actual length of the NDEF message in the value field, and
- The lock bits related to the memory area of the CC and the NDEF message are in the locked state.

In this state, the memory area is set to read-only, i.e., locked. This process is irreversible because setting the appropriate lock bits to 1 performs the transition from READ/WRITE to READ ONLY.

6.4.6 Determination of Life Cycle State

Before attempting a read or write operation, the NFC Forum Device in NFC Forum Reader/Writer Mode SHALL determine the state of a tag.

Generally, the most efficient approach is to read the complete tag contents and to buffer the data in its memory for analysis and parsing as follows:

- RID to capture HR0 to qualify it as an NDEF capable tag and to capture UID0-3
- RALL using UID0-3, to capture tag contents into local memory buffer
- Analyse CC and Lock Status bytes to determine the state

6.5 Rules for Life Cycle Operation

6.5.1 Detect NDEF on tag

Having determined the Life Cycle State of the tag as described in section 6.4, then the contents of the memory buffer SHALL be further analysed to detect the presence of a valid NDEF message as follows:

1. If byte 0 of block 1 is equal to E1h and byte 1 describes the right version number (see section 6.2) and the most significant nibble of byte 3 is equal to 0_h then go to item 2. Otherwise no NDEF data is detected in the Type 1 tag.
2. Parse the static data area contents already in memory and Read the dynamic memory data areas if relevant.

6.5.2 Read NDEF Message

The rules for how an NFC Forum Device in NFC Forum Reader/Writer Mode SHALL operate for the “read” context are as follows:

INITIALISED

No read of NDEF message is possible.

READ/WRITE

The memory contents SHALL be parsed to pass on the NDEF message to the application, if relevant then the dynamic memory data areas SHALL also be read.

READ ONLY

The memory contents SHALL be parsed to pass on the NDEF message to the application, if relevant then the dynamic memory data areas SHALL also be read.

Write NDEF Message

The rules for how an NFC Forum Device in NFC Forum Reader/Writer Mode SHALL operate for the “write” context are as follows:

INITIALISED

The writing of a new NDEF message SHALL occur as follows:

Write NMN = 00_h to ensure indication that no validate NDEF message is present during writing to allow detection of tear in the event that the tag is removed from the field prior to completion of operation.

Write VNo and RWA if required

Write NDEF Message TLV

Write NDEF Message data

Write NMN = E1_h as the last byte to be written in order to allow detection of tear in the event that the tag is removed from the field prior to completion of operation.

READ/WRITE

The over-writing of a new NDEF message SHALL occur as follows:

Write NMN = 00_h to invalidate existing NDEF message during writing to allow detection of tear in the event that the tag is removed from the field prior to completion of operation.

Write VNo and RWA if required

Write NDEF Message TLV

Write NDEF Message data

Write NMN = E1_h as the last byte to be written in order to allow detection of tear in the event that the tag is removed from the field prior to completion of operation.

READ ONLY

Write of NDEF message is not possible.

A. Appendix A

A.1 Example NDEF Mapping (Static Memory Model)

The contents of this appendix are only considered to be informative.

The following example NDEF message, which is copied from the Smartposter RTD draft specification document, has a total length of 23 bytes, ($=17_h$).

Table 15: Example Smartposter NDEF Message

Offset	Content	Length	Explanation
0	0xD1	1	NDEF header. TNF = 0x01 (Well Known Type). SR=1, MB=1, ME=1
1	0x02	1	Record name length (2 bytes)
2	0x12	1	Length of the Smart Poster data (18 bytes)
3	"Sp"	2	The record name
5	0xD1	1	NDEF header. TNF = 0x01, SR=1, MB=1, ME=1
6	0x01	1	Record name length (1 byte)
7	0x0E	1	The length of the URI payload (14 bytes)
8	"U"	1	Record type: "U"
9	0x01	1	Abbreviation: "http://www."
10	"nfc-forum.org"	13	The URI itself.

After mapping onto the NFC Forum Type 1 tag the example NDEF message of table 16, would look like the memory map of Figure 10 below.

HR0	HR1
11 _h	XX _h

EEPROM Memory Map									
Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
0	UID-0	UID-1	UID-2	UID-3	UID-4	UID-5	UID-6		Locked
1	CC0 (NMN) =E1 _h	CC1 (Vno) =10 _h	CC2 (TMS) =0E _h	CC3 (RWA) =00 _h	NDEF Message TLV T=03 _h	NDEF Message TLV L=17 _h	D1	02	Yes
2	12	'S'	'p'	D1	01	0E	'U'	01	Yes
3	'n'	'f'	'c'	'-'	'f'	'o'	'r'	'u'	Yes
4	'm'	'.'	'o'	'r'	'g'	Terminator TLV T=FE _h			Yes
5									Yes
6									Yes
7									Yes
8									Yes
9									Yes
A									Yes
B									Yes
C									Yes
D									Locked
E	LOCK-0	LOCK-1	OTP-0	OTP-1	OTP-2	OTP-3	OTP-4	OTP-5	

Figure 10: Memory Map of Example Smartposter NDEF Message

A.2 Example NDEF Mapping (Dynamic Memory Model)

The contents of this appendix are considered to be informative.

Figure 11 shows an example Type 1 tag with a dynamic memory of 32 blocks ($k=1F_h=31$).

There is no NDEF Message present in this example.

HR0	HR1
12 _h	xx _h

EEPROM Memory Map										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
UID	0	UID-0	UID-1	UID-2	UID-3	UID-4	UID-5	UID-6		Locked
Data	1	CC0 (NMN) =00 _h	CC1 (VNo) =10 _h	CC2 (TMS) =1F _h	CC3 (RWA) =00 _h	Lock Control TLV T=01 _h	Lock Control TLV L=03 _h	Lock Control TLV V0=F0 _h	Lock Control TLV V1=10 _h	Yes
Data	2	Lock Control TLV V2=33 _h	Memory Control TLV T=02 _h	Memory Control TLV L=03 _h	Memory Control TLV V0=F2 _h	Memory Control TLV V1=06 _h	Memory Control TLV V2=30 _h			Yes
Data	3									Yes
Data	4									Yes
Data	5									Yes
Data	6									Yes
Data	7									Yes
Data	8									Yes
Data	9									Yes
Data	A									Yes
Data	B									Yes
Data	C									Yes
Reserved	D									
Lock/Reserved	E	LOCK-0	LOCK-1	OTP-0	OTP-1	OTP-2	OTP-3	OTP-4	OTP-5	
Lock/Reserved	F	LOCK-2	LOCK-3	Reserved -0	Reserved -1	Reserved -2	Reserved -3	Reserved -4	Reserved -5	
Data	10									Yes
Data	11									Yes
Data	12									Yes
Data	13									Yes
Data	14									Yes
Data	15									Yes
Data	16									Yes
Data	17									Yes
Data	18									Yes
Data	19									Yes

EEPROM Memory Map										
Type	Block No.	Byte-0 (LSB)	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7 (MSB)	Lockable
Data	1A									Yes
Data	1B									Yes
Data	1C									Yes
Data	1D									Yes
Data	1E									Yes
Data	1F									Yes

Figure 11: Example Dynamic Memory Map

The tag is in INITIALISED state, the main memory area is set as following:

- All lock bits are set to 0b: Lock0 = Lock1 = Lock2 = Lock3 = 00h
- The CC is set as follows:
 - CC0 = 00h to indicate that NDEF data is not present.
 - CC1 = 10h to indicate support of the version 1.0 (major number 1_h, minor number 0h) of this operation document,
 - CC2 = 1Fh to indicate 32 blocks or 256 bytes of memory size,
 - CC3 = 00h to indicated read and write access granted without any security.
- The data area contains four TLV blocks in the following order:
 - Lock Control TLV:

T = 01h

L = 03h

V = F0 10 33h indicates that each lock bit locks 1 page, each page is 8 bytes, and the lock area is 16 bits long starting at the byte address 120 as calculated by the formula;

ByteAddr = PageAddr * 2^{BytesPerPage} + ByteOffset = 15 * 2³ + 0 = 120 where:

- Position = F0h contains PageAddr = Fh = 15 and ByteOffset = 0h
- Size = 10h = 16 bits
- PageControl = 33h contains BytesPerPage = 3h (2³ = 8 bytes) and BytesLockedPerLockBit = 3h (2³ = 8 bytes).

- Reserved Memory Control TLV:

T = 02h

L = 03h

V = F20630h indicates that the reserved area is 6 bytes long starting at the byte address 122 as calculated by the formula;

$$\text{ByteAddr} = \text{PageAddr} * 2^{\text{BytesPerPage}} + \text{ByteOffset} = 15 * 2^3 + 2 = 122$$

where:

- Position = F2h contains PageAddr = Fh = 15 and ByteOffset = 2h
- Size = 06h
- PageControl = 30h contains BytesPerPage = 3h, and least significant nibble = 0h (Ignored)

B. Revision History

The following table outlines the revision history of Type 1 Tag Operation Specification.

Table 16: Revision History

Document Name	Revision and Release Date	Status	Change Notice	Supersedes
NFCForum-TS-Type-1-Tag_1.0	1.0, July 2007	Final	None	



Type 2 Tag Operation

Technical Specification

NFC Forum™

NFCForum-TS-Type-2-Tag_1.0

2007-07-09

RESTRICTIONS ON USE

This specification is copyright © 2005-2007 by the NFC Forum, and was made available pursuant to a license agreement entered into between the recipient (Licensee) and NFC Forum, Inc. (Licensor) and may be used only by Licensee, and in compliance with the terms of that license agreement (License). If you are not the Licensee, you are not authorized to make any use of this specification. However, you may obtain a copy at the following page of Licensor's Website: http://www.nfc-forum.org/resources/spec_license after entering into and agreeing to such license terms as Licensor is then requiring. On the date that this specification was downloaded by Licensee, those terms were as follows:

1. LICENSE GRANT.

Licensor hereby grants Licensee the right, without charge, to copy (for internal purposes only) and share the Specification with Licensee's members, employees and consultants (as appropriate). This license grant does not include the right to sublicense, modify or create derivative works based upon the Specification.

2. NO WARRANTIES.

THE SPECIFICATION IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY, COMPLETENESS AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL LICENSOR, ITS MEMBERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE SPECIFICATION.

3. THIRD PARTY RIGHTS.

Without limiting the generality of Section 2 above, LICENSOR ASSUMES NO RESPONSIBILITY TO COMPILE, CONFIRM, UPDATE OR MAKE PUBLIC ANY THIRD PARTY ASSERTIONS OF PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS THAT MIGHT NOW OR IN THE FUTURE BE INFRINGED BY AN IMPLEMENTATION OF THE SPECIFICATION IN ITS CURRENT, OR IN ANY FUTURE FORM. IF ANY SUCH RIGHTS ARE DESCRIBED ON THE SPECIFICATION, LICENSOR TAKES NO POSITION AS TO THE VALIDITY OR INVALIDITY OF SUCH ASSERTIONS, OR THAT ALL SUCH ASSERTIONS THAT HAVE OR MAY BE MADE ARE SO LISTED.

4. TERMINATION OF LICENSE.

In the event of a breach of this Agreement by Licensee or any of its employees or members, Licensor shall give Licensee written notice and an opportunity to cure. If the breach is not cured within thirty (30) days after written notice, or if the breach is of a nature that cannot be cured, then Licensor may immediately or thereafter terminate the licenses granted in this Agreement.

5. MISCELLANEOUS.

All notices required under this Agreement shall be in writing, and shall be deemed effective five days from deposit in the mails. Notices and correspondence to the NFC Forum address as it appears below. This Agreement shall be construed and interpreted under the internal laws of the United States and the Commonwealth of Massachusetts, without giving effect to its principles of conflict of law.

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

Contents

1	Introduction.....	1
1.1	Objectives.....	1
1.2	Purpose.....	1
1.3	Applicable Documents or References	1
1.4	Administration.....	1
1.5	Special Word Usage	2
1.6	Name and Logo Usage	2
1.7	Intellectual Property	2
1.8	Glossary	2
1.9	Convention and notations	3
1.9.1	Representation of numbers	3
2	Memory Structure and Management.....	4
2.1	Static Memory Structure.....	5
2.1.1	Unique Identifier (UID).....	6
2.1.2	Internal Bytes.....	6
2.1.3	Static Lock bytes.....	6
2.1.4	Capability Container	6
2.1.5	Data Area for Static Memory Structure	6
2.2	Dynamic Memory Structure	7
2.2.1	Reserved Bytes	8
2.2.2	Static and Dynamic Lock Bits	8
2.2.3	Data Area for Dynamic Memory Structure	9
2.3	TLV blocks.....	10
2.3.1	Lock Control TLV	11
2.3.2	Memory Control TLV.....	12
2.3.3	NDEF Message TLV	13
2.3.4	Proprietary TLV.....	14
2.3.5	NULL TLV.....	14
2.3.6	Terminator TLV.....	14
3	RF Interface.....	15
4	Framing / Transmission Handling.....	16
5	Command Set	17
5.1	Tag Commands and Responses Set.....	17
5.1.1	READ	17
5.1.2	WRITE.....	18
5.1.3	SECTOR SELECT	19
5.1.4	ACK and NACK.....	21
6	NDEF Detection and Access.....	22
6.1	NDEF Management.....	22
6.1.1	Version Treating	23
6.2	NDEF Storage	24
6.3	Life Cycle	24
6.3.1	INITIALISED State.....	24
6.3.2	READ/WRITE State.....	24
6.3.3	READ-ONLY State	25
6.4	Command Sequence Description.....	25

6.4.1	NDEF Detection Procedure	25
6.4.2	NDEF Read Procedure.....	26
6.4.3	NDEF Write Procedure.....	26
6.4.4	State Changes.....	27

Figures

Figure 1: Static Memory Structure.	5
Figure 2: UID Coding.....	6
Figure 3: Example of Dynamic Memory Structure.	7
Figure 4: Length Field Formats	10
Figure 5: Timing Overview READ Command.....	18
Figure 6: Timing Overview WRITE Command.....	19
Figure 7: Timing Overview SECTOR SELECT Command	20
Figure 8: Life Cycle with State Changes (transitions)	28
Figure 9: Example of Static Memory Structure.....	31
Figure 10: Example of Dynamic Memory Structure.	32

Tables

Table 1: Defined TLV blocks.....	11
Table 2: Command Set overview	17
Table 3: READ Command	17
Table 4: WRITE Command.....	18
Table 5: SECTOR SELECT Command	20
Table 6: ACK and NACK	21
Table 7: Example of coding of the CC bytes of block 3	23
Table 8: Handling of the mapping document version numbers.....	23
Table 9: Revision History.....	44

1 Introduction

This specification is part of the NFC Forum documentation about tag types that an NFC Forum device needs to support in reader/writer mode.

This specification documents how an NFC Forum Device SHALL operate an NFC Forum Type 2 tag platform. This is not a specification of the NFC Forum Type 2 tag platform itself.

1.1 Objectives

The purpose of this specification is to document the requirements and to specify, with a set of rules and guidelines, the NFC Forum Device operation and management of the Type 2 tag platform.

This specification assumes that the Collision Detection and Device Activation activities have been performed as documented in the Mode Switch specifications [DIGPROT] and [ANINT].

This specification also defines the data mapping and how the NFC Forum Device detects, reads, and writes NDEF data into the Type 2 tag platform in order to achieve and maintain interchangeability and interoperability.

1.2 Purpose

The purpose of this specification is to document the requirements and to specify, with a set of rules and guidelines, the NFC Forum Device operation and management of a Type 2 tag platform.

This specification also defines the data mapping and how the NFC Forum Device detects, reads, and writes NDEF data into the Type 2 tag platform in order to achieve and maintain interchangeability and interoperability.

1.3 Applicable Documents or References

[ANINT]	NFC Analogue Interface Specification.
[DIGPROT]	NFC Digital Protocol specification.
[NDEF]	"NFC Data Exchange Format (NDEF)" NFC Forum™, May 2006.
[RFC 2119]	S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Harvard University, March 1997.

1.4 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.

1.5 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 Type 2 Tag Operation 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

CC

Capability Container

LSB

least significant byte

lsb

least significant bit

Mandatory NDEF Message TLV, first NDEF Message TLV

NDEF Message TLV detected by the NDEF detection procedure

MSB

most significant byte

msb

most significant bit

NDEF

NFC Data Exchange Format (see [NDEF])

NFC Forum device

A device that supports the following modus operandi: Initiator, Target and Reader/Writer. It may also support Card Emulator. NFC Forum compliant Device. In this document the NFC Forum device is always using the Reader/Writer modus operandi (for more information see [DIGPROT]).

RF

Radio Frequency

RFU

Reserved for future use

Type 2 tag platform

A legacy platform supporting a subset of a Technology (also called Technology Subset). Type 2 tag platform, which uses a particular subset of NFC – Type A technology including anticollision (for more information see [DIGPROT]).

UID

Unique identifier

1.9 Convention and notations

1.9.1 Representation of numbers

The following conventions and notations apply in this document unless otherwise stated.

- Binary numbers are represented by strings of digits 0 and 1 shown with the most significant bit (msb) left and the least significant bit (lsb) right, “b” is added at the end.
Example: 11110101b
- Hexadecimal numbers are represented using the numbers 0 - 9 and the characters A – F, an “h” is added at the end. The most significant byte (MSB) is shown on the left, the least significant byte (LSB) on the right.
Example: F5h
- Decimal numbers are represented as is (without any tailing character).
Example: 245

2 Memory Structure and Management

Type 2 tag platform is based on a particular memory chip with a certain memory size and space for data. The following sections describe the details of such memory chip, and in particular its memory structure and management.

The memory structure (or layout) depends on the memory size of the tag:

- A static memory structure is used for tag with memory size equal to 64 bytes, and
- A dynamic memory structure is used by tag with memory size bigger than 64 bytes.

The memory structure is divided in blocks containing 4 bytes each (see Figure 1 and Figure 3). Each block is numbered from 0 to 15 for static memory structure or from 0 to k for dynamic memory structure. The number associated to a block is also called block number. The 4 bytes inside each block are numbered from 0 to 3. For each block byte 0 is the MSB and byte 3 is the LSB. Byte 0 of block 0 indicates the MSB. Byte 3 of block 15 for static memory structure or byte 3 of block k for dynamic memory structure indicates the LSB.

The blocks are grouped in sectors. The sector is defined as 256 contiguous blocks (1024 bytes, or 1KB).

In the definition of this document the bit and byte ordering when defining packets and messages follows the big-endian byte order.

The next two sections described in details the two memory structures (also called layouts).

2.1 Static Memory Structure

This memory structure is used by Type 2 tag platform with a physical memory size equal to 64 bytes. Figure 1 shows the memory layout of such tag. It is composed of different fields:

- UID, Unique identifier (see section 0),
- Internal, reserved bytes for manufacturing usage (see section 2.1.2),
- Lock, static lock bytes (see section 2.1.3) to switch the tag from READ/WRITE state to READ-ONLY state (see section 6.3),
- CC, Capability Container bytes (see section 2.1.4),
- Data, bytes used to store information (see section 2.1.5).

Byte Number	0	1	2	3	Block
UID / Internal	UID0	UID1	UID2	Internal0	0
Serial Number	UID3	UID4	UID5	UID6	1
Internal / Lock	Internal1	Internal2	Lock0	Lock1	2
CC	CC0	CC1	CC2	CC3	3
Data	Data0	Data1	Data2	Data3	4
Data	Data4	Data5	Data6	Data7	5
Data	Data8	Data9	Data10	Data11	6
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	..	..	..	15

Figure 1: Static Memory Structure.

2.1.1 Unique Identifier (UID)

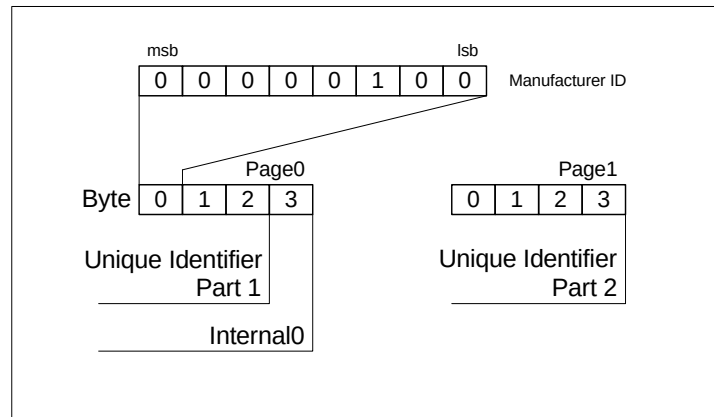


Figure 2: UID Coding

The 7 bytes unique identifier (UID0-6, see Figure 1) are contained in bytes 1-3 of block 0, and the 4 bytes of block 1. The UID bytes are write-protected after having been programmed by the IC manufacturer.

For more information about UID see Type 2 tag section in [DIGPROT].

2.1.2 Internal Bytes

These bytes are reserved for manufacturing use. The NFC Forum device SHALL NOT use them to store information data.

2.1.3 Static Lock bytes

The bits of byte 2 and 3 of block 2 represent the field-programmable read-only locking mechanism called static lock bytes. Depending on the value of the bits of the static lock bytes two configurations are possible:

- All bits are set to 0b, the CC area and the data area of the tag can be read and written.
- All bits are set to 1b, the CC area and the data area of the tag can be only read.

The locking bits are set to 1b via a standard write command to block 2 (BNo = 2, see section 5.1.2). To set all static lock bits to 1b, the NFC Forum device SHALL set bytes D2 and D3 of the WRITE command to FFh and set the remaining two bytes D0 and D1 to any value.

This process is irreversible: if one bit of the lock bytes is set to 1b, it cannot be changed back to 0b.

2.1.4 Capability Container

The Capability Container (CC) manages the information of the Type 2 tag platform. The four bytes of block 3 contain the so called CC. A detailed description of the CC is given in chapter 6.1.

2.1.5 Data Area for Static Memory Structure

Block 4 to 15 is the available data area for information storage. The NFC Forum device SHALL write the data area consecutively in order starting from byte 0 of block 4 up to byte 3 of block 15.

For static memory structure the data area size is equal to 48 bytes (see also CC description in section 6.1).

2.2 Dynamic Memory Structure

This memory structure (or layout) is applied to Type 2 tag platform with a memory size bigger than 64 bytes. Figure 3 shows an example of memory layout of such tag. It is composed of different fields:

- UID, Unique identifier as defined in section 0,
- Internal, bytes for manufacturing usage as defined in section 2.1.2,
- Reserved, reserved bytes (see section 2.2.1),
- Lock, static and dynamic lock bytes (see section 2.2.2) to switch the tag from READ/WRITE state to READ-ONLY state (see section 6.3),
- CC, Capability Container bytes (see section 2.1.4),
- Data, bytes used to store information (see section 2.2.3).

Byte Number	0	1	2	3	Block
UID / Internal	UID0	UID1	UID2	Internal0	0
Serial Number	UID3	UID4	UID5	UID6	1
Internal / Lock	Internal1	Internal2	Lock0	Lock1	2
CC	CC0	CC1	CC2	CC3	3
Data	Data0	Data1	Data2	Data3	4
Data	Data4	Data5	Data6	Data7	5
Data	Data8	Data9	Data10	Data11	6
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	..	..	..	n
Lock / Reserved	..	..	..	..	.
Lock / Reserved	..	..	..	..	.
Lock / Reserved	..	..	..	..	k

Figure 3: Example of Dynamic Memory Structure.

In Figure 3 each block is numbered from 0 to k. The block n indicates the last block of the data area. Blocks from n+1 to k contains reserved or lock bytes.

NOTE Dynamic lock bytes and reserved bytes might be located at any byte address in between or at the end of the data areas starting from block 16. A more detailed example of dynamic memory structure is shown in Appendix B.

Compared to the static memory structure, the dynamic memory structure might contain optional configuration information to describe details of dynamic lock bits, and to identify reserved memory areas in the data area using the Lock Control TLV and the Memory Control TLV (see section 2.3).

2.2.1 Reserved Bytes

The reserved bytes belong to reserved memory areas. The NFC Forum device SHALL ignore and jump over the reserved bytes during read and write operations. Reserved bytes are identified by one or more Memory Control TLV blocks, see section 2.3.2.

2.2.2 Static and Dynamic Lock Bits

A tag with a dynamic memory structure contains two kinds of lock bits:

6. Static lock bits as specified in section 2.1.3, and
7. Dynamic lock bits described later on.

The dynamic lock bits are called “dynamic” because their position(s) inside the tag can change. This is in contrast to the static lock bytes as their position is fixed (see section 2.1.3).

Lock areas are only needed on Type 2 tag platform that allows the transition from READ/WRITE state to READ-ONLY state (see section 6.4.4). The NFC Forum device SHALL ignore and jump over the bytes that belong to lock areas during read operations of NFC Forum data inside Type 2 tag platform in READ-ONLY state (see section 6.3).

The default settings of the dynamic lock bits are:

1. *The position of the dynamic lock bits* starts from the first byte after the data area (see section 2.2.3, and Figure 3). The bytes that contain the dynamic lock bits are called dynamic lock bytes.
2. *The number of dynamic lock bits* is equal to data area size minus 48 (in bytes) divided by 8. If the division result is not an integer number, the number of lock bytes is equal to the closest integer number that is bigger than the division result i.e.

$$\text{NumberOfDynamicLockBits} = \lceil (\text{DataAreaSize} - 48) / 8 \rceil$$

The number of dynamic lock bytes is equal to

$$\text{NumberOfDynamicLockBytes} = \lceil (\text{DataAreaSize} - 48) / 64 \rceil$$

When the number of the dynamic lock bits is not a multiple of 8, the last dynamic lock byte is partially filled with these bits. In this byte the dynamic lock bits are located starting from the lsb to the msb. The part of the byte that does not contain dynamic lock bits, is filled with reserved bits that the NFC Forum device SHALL always set to 0b.

The NFC Forum device SHALL overrule the default settings of the dynamic lock bits when one or more Lock Control TLV blocks are present (see section 2.3). The NFC Forum device SHALL calculate the position and the number of the dynamic lock bits from the information contained in the Lock Control TLV.

Type 2 tag platforms that are delivered in READ-ONLY state can indicate the lock areas as reserved memory areas. This allows to use one Memory Control TLV (see section 2.3) to indicate contiguous and alternating lock areas, and reserved areas.

Depending on the values of the static and dynamic lock bits two configurations are possible:

- All bits are set to 0b, the CC area and the data area of the tag can be read and written.
- All bits are set to 1b, the CC area and the data area of the tag can be only read.

To set to 1b static lock bits see section 2.1.3.

The NFC Forum device SHALL set the dynamic locking bits to 1b via a standard WRITE command (see section 5.1.2). Being the write command block-wise the NFC Forum device SHALL set to 1b only the bits that belong to the dynamic lock bits of the block.

If a block contains one or more dynamic lock bytes and one or more non-lock bytes, the NFC Forum device MAY send first an READ command (see section 5.1.1) and then a WRITE command on the same block. From the response of the READ command the values of the non-lock bytes are retrieved. The NFC Forum device MAY use these values in the WRITE command to avoid changing the value of the non-lock bytes, and to set to 1b the dynamic lock bits.

The setting of the static and dynamic lock bits is irreversible: if one bit lock bit is set to 1b, it cannot be changed back to 0b.

2.2.3 Data Area for Dynamic Memory Structure

The data area for dynamic memory structure is contained from block 4 up to the last block of the memory including the 48 bytes of the static memory structure (see section 2.1), and excluding dynamic lock bytes and reserved bytes. The data area is the only memory area where the NFC Forum device SHALL read and write the TLV blocks (see section 2.3). The NFC Forum device SHALL write the data area sequentially starting from byte 0 of block 4 to byte 3 of block k jumping over dynamic lock bytes and reserved bytes. The data area size in bytes is equal to:

$$4 \cdot (k - 3) - \text{DynamicLockBytes} - \text{ReservedBytes}$$

The previous calculation includes the data area of the static memory structure equal to 48 bytes (see section 2.1), and supposes that the first block is numbered starting from 0 i.e. block 0. The value k indicates the overall number of blocks that belong to one or more sector reduced by 1. E.g. a dynamic memory structure composed of 2 sectors has 512 blocks; hence k is equal to 511. For dynamic memory structure k is bigger than 15 i.e. $k > 15$.

Compared to the static memory structure the dynamic memory structure adds a number of data area bytes equal to:

$$4 \cdot (k - 3) - \text{DynamicLockBytes} - \text{ReservedBytes} - 48$$

The memory starting from block 16 contains all dynamic lock bytes, all reserved bytes (see also NOTE on page 7), and all data area bytes added by the dynamic memory structure.

2.3 TLV blocks

A TLV block consists of one to three fields:

- *T* (tag field or T field) identifies the type of the TLV block, see Table 1, and consists of a single byte encoding a number from 00h to FFh. The tag values 04h to FCh and FFh are reserved for future use by the NFC Forum.
- *L* (length field or L field) provides the size in bytes of the value field. It has two different formats composed of one, or three bytes. The NFC Forum device SHALL understand both two length field formats. Figure 4 shows the two different length field structures. However, depending on the tag field value, the length field may not be present.
 - *One byte format*: The NFC Forum device SHALL use the one byte format to code the length of the value field between 00h and FEh bytes. The NFC Forum device SHALL interpret this byte as a cardinal if the value is between 00h and FEh. If it contains FFh, the NFC Forum device SHALL interpret the value as flag that specifies that the length field is composed of more than one byte.
 - *Three consecutive bytes format*: The NFC Forum device SHALL use this format to code the length of the value field between 00FFh and FFFEh bytes. The first byte is assumed to be a flag equal to FFh indicating that two more bytes are present. The NFC Forum device SHALL interpret the two more bytes as a word. The NFC Forum device SHALL interpret this word as a cardinal if the value is between 00FFh and FFFEh. The value FFFFh is reserved for future use (RFU).

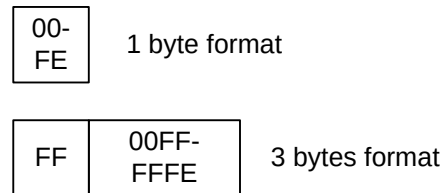


Figure 4: Length Field Formats

- *V* (value field, or V field) If the length field is equal to 00h or there is no length field, the value field is not present, i.e. the TLV block is empty. If there is the length field and it indicates a length N bigger than zero ($N > 0$), the value field consists of N consecutive bytes.

Table 1 lists the TLV blocks specified by this document that are described in the following sections.

Table 1: Defined TLV blocks

<i>TLV block name</i>	<i>Tag Field Value</i>	<i>Short Description</i>
NULL TLV	00h	It might be used for padding of memory areas and the NFC Forum device SHALL ignore this
Lock Control TLV	01h	It defines details of the lock bits
Memory Control TLV	02h	It identifies reserved memory areas
NDEF Message TLV	03h	It contains an NDEF message, see [NDEF]
Proprietary TLV	FDh	Tag proprietary information
Terminator TLV	FEh	Last TLV block in the data area

The NFC Forum device SHALL write the TLV blocks in a specific order inside the data area (see section 2.1.5, and 2.2.3) following the rules below:

- NDEF Message TLVs and Proprietary TLVs are present after all Lock Control TLVs and Memory Control TLVs.
- If present the Terminator TLV is the last TLV block on the Type 2 tag platform.

NULL TLV and Terminator TLV are the only TLV blocks that are 1 byte long (i.e. composed of only the Tag field, see below).

NFC Forum devices SHALL ignore and jump over those TLV blocks that make use of reserved tag field values. To jump over a TLV block with reserved tag field values, the NFC Forum device SHALL read the length field to understand the length of the value field.

NOTE Future definitions of TLV blocks composed of only the tag field are not backward compatible with this NFC Forum specification.

2.3.1 Lock Control TLV

The Lock Control TLV can be present inside the Type 2 tag platform. An NFC Forum device SHALL be able to read and process it. The Lock Control TLV provides control information about the lock areas where the dynamic lock bytes are located (see section 2.2.2). Each Lock Control TLV indicates a single lock area. More lock areas are indicated using more Lock Control TLV blocks. Below the encoding of the 3 TLV fields of the Lock Control TLV are shown:

- T is equal to 01h (see Table 1).
- L is equal to 03h.
- V is composed of 3 bytes that uniquely identify the position and the size of the lock area, and the number of bytes locked by each bit of the dynamic lock bytes. The 3 bytes are encoded in the following way:
 - Position, MSB. It codes the position inside the tag of the lock area. The position byte consists of 2 parts (to calculate the bytes address from the position byte see below):
 - PagesAddr, most significant nibble (4 bits), coded as number of pages (0h=0...Fh=15) and

- ByteOffset, least significant nibble, coded as number of bytes (0h=0...Fh=15).
- Size, middle byte, coded as number of bits (01h=1...FFh=255, 00h=256). It indicates the size in bits of the lock area i.e. the number of dynamic lock bits. If the number of dynamic lock bits is not a multiple of 8, they are stored inside the dynamic lock bytes as explained in the description of the default setting of the dynamic lock bits (see section 2.2.2).
- Page control, LSB. The page control provides general control information: the size in bytes of a page, and the number of bytes that each dynamic lock bit is able to lock. Page control byte is split up into two nibbles of 4 bits each:
 - BytesPerPage: least significant nibble, coded as 2^n (0h=RFU, 1h=1...Fh=15). It indicates the number of bytes per page.
 - BytesLockedPerLockBit: most significant nibble, coded as 2^n (0h=RFU, 1h=1...Fh=15). It indicates the number of bytes that each dynamic lock bit is able to lock.

The NFC Forum device SHALL calculate the byte address (ByteAddr) of the beginning of the lock area in the following way:

$$\text{ByteAddr} = \text{PageAddr} \cdot 2^{\text{BytesPerPage}} + \text{ByteOffset}$$

The ByteAddr is calculated from the beginning of the overall memory tag; Byte 0 of Block 0 is indicated by ByteAddr equal to 0.

The ByteAddr are used to read and write the relative lock area using the appropriate READ and WRITE commands (see chapter 5). The page definition has nothing to do with the block definition used by READ and WRITE commands.

An example of use of the BytesLockedPerLockBit is e.g. if the memory area loocked by a single dynamic lock bit is 8 bytes, the BytesLockedPerLockBit is equal to 3 i.e. $2^{\text{BytesLockedPerLockBit}} = 2^3 = 8$ bytes.

NOTE The Lock Control TLV might be skipped if a Type 2 tag platform is in READ-ONLY state (see section 6.3). Lock Control TLV blocks can be replaced by Memory Control TLV indicating the same memory areas for Type 2 tag platform in READ-ONLY state (see section 2.3.2).

2.3.2 Memory Control TLV

The Memory Control TLV can be present inside the Type 2 tag platform, and an NFC Forum device SHALL be able to read and process it. It provides control information about the reserved areas where the reserved bytes are located (see section 2.2.1), and the size of the reserved bytes.

If the Type 2 tag platform is delivered by the vendors in READ-ONLY state (see section 6.3), the NFC Forum device MAY use the Memory Control TLV to indicate control information for reserved and lock areas. Contiguous and alternating lock and reserved areas MAY be indicated by a single Memory Control TLV.

Below the encoding of the 3 TLV fields of Memory Control TLV are shown:

- T is equal to 02h (see Table 1).
- L is equal to 03h.
- V is composed of 3 bytes that uniquely identifies the position and the size of the reserved area. The 3 bytes are encoded in the following way:
 - Position, MSB. It codes the position inside the tag of the reserved area. The Position byte consists of 2 parts (to calculate the bytes address from the position byte see below):
 - PagesAddr, most significant nibble, coded as number of pages (0h=0...Fh=15) and
 - ByteOffset, least significant nibble, coded as number of bytes (0h=0...Fh=15).
 - Size, middle byte, coded as number of bytes (1h=1, FFh=255, 0h=256). It indicates the size in bytes of the reserved area.
 - Partial Page Control, LSB. The partial page control provides the size in bytes of a page. It is split up into two nibbles of 4 bits each:
 - BytesPerPage nibble: least significant nibble, coded as 2^n (0h=RFU, 1h=1...Fh=15). It indicates the number of bytes per page.
 - Most significant nibble is RFU.

The NFC Forum device SHALL calculate the byte address (ByteAddr) of each reserved area in the following way:

$$\text{ByteAddr} = \text{PageAddr} \cdot 2^{\text{BytesPerPage}} + \text{ByteOffset}$$

The ByteAddr is calculated from the beginning of the overall memory tag; Byte 0 of Block 0 is indicated by ByteAddr equal to 0.

The page definition has nothing to do with the block definition used by READ and WRITE commands (see section 5.1).

2.3.3 NDEF Message TLV

The NDEF Message TLV is always present inside the Type 2 tag platform. It stores the NDEF message inside the Value field (see [NDEF]). The NFC Forum device SHALL be able to read and process the first (or mandatory) NDEF message (see section 6.4.1); anyhow further NDEF Message TLV blocks can be present. The always present mandatory NDEF Message TLV provides the starting point when writing the NDEF Message into the Type 2 tag. I.e. an NDEF Message cannot be written before the NDEF Message TLV, this avoids corrupting the possible Memory and Lock Control TLVs (see section 6.4.3). Below the encoding of the 3 TLV fields of NDEF Message TLV is shown:

- T is equal to 03h (see Table 1).
- L is equal to the size in bytes of the stored NDEF message.
- V stores the NDEF message (see [NDEF]).

An empty NDEF Message TLV is defined as an NDEF Message TLV with L field equal to 00h, and no V field (i.e. no NDEF message is present in the V field, see [NDEF]).

A non-empty NDEF Message TLV can contain either empty or non-empty NDEF messages. The definition of empty NDEF message is given in Appendix A.

2.3.4 Proprietary TLV

The Proprietary TLV contains proprietary information. A Type 2 tag platform contains zero, one or more Proprietary TLV. The NFC Forum device MAY ignore the data contained in this TLV block. Below the encoding of the 3 TLV fields of Proprietary TLV are shown:

- T is equal to FDh (see Table 1).
- L is equal to the size in bytes of the proprietary data in the Vale field.
- V contains any proprietary data.

2.3.5 NULL TLV

The Null TLV can be used for padding of the data area. A Type 2 tag platform contains zero, one or more NULL TLV. The NFC Forum device SHALL ignore and jump over this TLV block. NULL TLV is composed of 1 byte tag field. Below the encoding of the tag field of the NULL TLV are shown:

- T is equal to 00h (see Table 1).
- L is not present.
- V is not present.

2.3.6 Terminator TLV

The Terminator TLV can be present inside the Type 2 tag platform, and an NFC Forum device SHALL be able to read and process it. The Terminator TLV is the last TLV block in the data area. Terminator TLV is composed of 1 byte tag field. Below the encoding of the tag fields of the Terminator TLV are shown:

- T is equal to FEh (see Table 1).
- L is not present.
- V is not present.

3 RF Interface

The RF interface of the NFC Forum device is defined in [ANINT]. The NFC Forum device SHALL comply with the RF interface as defined in the relevant clauses of [ANINT].

4 Framing / Transmission Handling

This chapter describes the framing (also called packet structures), and the transmission handling of the NFC Forum device.

The NFC Forum device SHALL comply with the bit coding, the character coding, the frame coding, the byte coding and as well the commands and responses up to and including the activation sequence of the Type 2 tag as defined in [DIGPROT] (please note that the activation is implicit).

For more information about commands and responses, see chapter 5.

5 Command Set

This chapter describes the command set of the NFC Forum device to communicate to the Type 2 tag platform. It provides the basis used to detect NDEF data, and the read and write access to the NDEF data.

5.1 Tag Commands and Responses Set

Table 2 below shows the tag responses for each specific command.

Table 2: Command Set overview

<i>Command</i>	<i>Response</i>	<i>ISO 14443 compliance</i>
READ	16 bytes data, NACK	Low Level Access Command
WRITE	ACK, NACK	Low Level Access Command
SECTOR SELECT	Passive ACK	Low Level Access Command

The commands are outlined in the following clauses.

5.1.1 READ

The NFC Forum device SHALL be able to send the READ command, and to receive the 16 bytes data response or the NACK.

Table 3 describes the READ commands and the relative responses. The parity integrity mechanism (i.e. CRC and parity) is provided by the frame related to the READ command, see section 4.

Table 3: READ Command

	<i>Name</i>	<i>Code (1 byte)</i>	<i>Parameter</i>	<i>Data</i>	<i>Integrity mechanism</i>
Command	READ	30h	BNo (1 byte): 00h - FFh	-	CRC, Parity
Response	16 bytes data	-	-	16 bytes data	CRC, Parity
	NACK	See section 5.1.4 (4 bits)	-	-	-

The READ command has the command code 30h and needs the block number (BNo) as a parameter. The block number is explained in chapter 2. The Type 2 tag platform responds to a READ command by sending 16 bytes starting from the block number defined in the READ command (e.g. if BNo is equal to 03h blocks 3, 4, 5, and 6 are returned), see Figure 5. The memory organization of the Type 2 tag platform is described in chapter 2. In case of error the Type 2 tag platform sends a NACK response, and goes to IDLE or HALT state (for more information see the SENSE and SLEEP state in [DIGPROT] respectively for IDLE or HALT state).

The READ command is unable to select the sector (see section 2.2). The READ command reads blocks that belong to the currently selected sector. To select a different sector, the SECTOR SELECT command is used.

To calculate the block number (BNo) from the byte address (ByteAddr, see section 2.3.1 and 2.3.2) the NFC Forum device SHALL use the following expression $BNo = \lfloor ByteAddr / 4 \rfloor$. If the formula gives as result BNo bigger than 255, the block indicated by BNo does not belong to sector 0 (default sector). In this case the NFC Forum device SHALL use the SECTOR SELECT command first to switch to the correct sector (SecNo) equal to $SecNo = \lfloor BNo / 256 \rfloor$. Then the NFC Forum device SHALL use the READ command with the adapted block number (BNo') instead of BNo. The NFC Forum device SHALL calculate BNo' from the formula $BNo' = BNo \bmod 256$. With Mod the module operation that gives the remainder of the division $ByteAddr / 256$.

Typical NFC Forum device timeout value for the READ command is 5msec. I.e. the NFC Forum device after sending a READ command SHALL wait up to 5msec for incoming responses from the Type 2 tag platform before triggering a timeout.

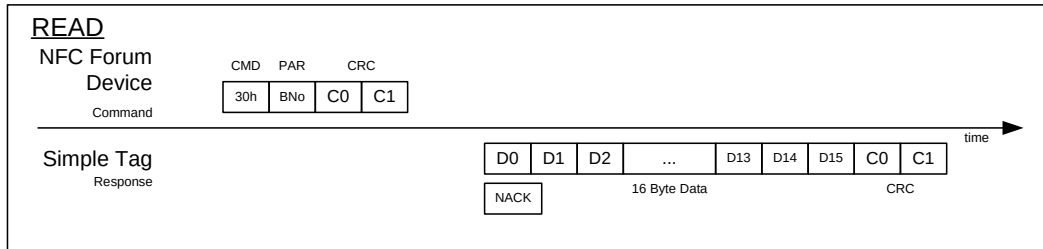


Figure 5: Timing Overview READ Command

5.1.2 WRITE

The NFC Forum device SHALL be able to send the WRITE command and to receive the ACK or NACK response.

Table 4 describes the WRITE commands and the relative responses ACK and NACK. The parity integrity mechanism (i.e. CRC and parity) is provided by the frame of the WRITE command, see section 4.

Table 4: WRITE Command

	Name	Code (1 byte)	Parameter	Data	Integrity mechanism
Command	WRITE	A2h	BNo (1 byte): 00h – FFh	4 bytes data	CRC, Parity
Response	ACK	See section 5.1.4 (4 bits)	-	-	-
	NACK	See section 5.1.4 (4 bits)	-	-	-

The WRITE command has the command code A2h followed by the block number (BNo) parameter. The NFC Forum device SHALL use the WRITE command for programming data.

This MAY be the CC bytes (see section 6.1), the lock bytes, or the data area bytes (see chapter 2). The NFC Forum device SHALL use the WRITE command block-wise, programming 4 bytes at once.

If the WRITE command is executed successfully by the Type 2 tag platform, the ACK response is sent back, see Figure 6. In case of error the Type 2 tag platform sends a NACK response, and goes to IDLE or HALT state (for more information see the SENSE and SLEEP state in [DIGPROT] respectively for IDLE or HALT state).

The WRITE operation is block-wise i.e. it writes always the 4 bytes of the whole block. The NFC Forum device SHALL write the new byte values, and overwrite with the same values the bytes that remain unchanged. The NFC Forum device MAY read the block first (i.e. READ operation), if the values of the bytes that remain unchanged are not known in advance.

The WRITE command is unable to select the sector (see section 2.2). The WRITE command reads blocks that belong to the currently selected sector. To select a different sector, the SECTOR SELECT command is used.

To calculate the block number (BNo) from the byte address (ByteAddr, see section 2.3.1 and 2.3.2) the NFC Forum device SHALL use the following expression $BNo = \lfloor ByteAddr / 4 \rfloor$. If the formula gives as result BNo bigger than 255, the block indicated by BNo does not belong to sector 0 (default sector). In this case the NFC Forum device SHALL use the SECTOR SELECT command first to switch to the correct sector (SecNo) equal to $SecNo = \lfloor BNo / 256 \rfloor$. Then the NFC Forum device SHALL use the WRITE command with the adapted block number (BNo') instead of BNo. To calculate BNo' the NFC Forum device SHALL use the following formula $BNo' = BNo \bmod 256$. With Mod the module operation that gives the remainder of the division $ByteAddr / 256$.

The NFC Forum device timeout value for the WRITE command is 10msec. I.e. the NFC Forum device after sending a WRITE command SHALL wait up to 10msec for incoming responses from the Type 2 tag platform before triggering a timeout.

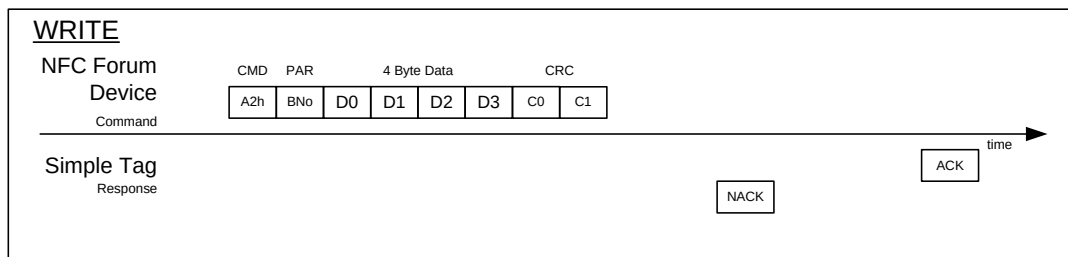


Figure 6: Timing Overview WRITE Command

5.1.3 SECTOR SELECT

In case the NFC Forum device supports Type 2 tag platform bigger than 1KB, it SHALL support the SECTOR SELECT command. In such case, the NFC Forum device SHALL be able to send the SECTOR SELECT command, and to receive the ACK and NACK responses.

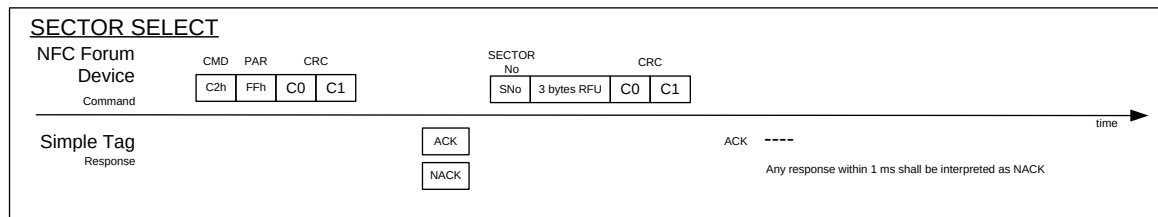
Table 5 describes the SECTOR SELECT command divided into 2 packets and the relative responses ACK, passive ACK and NACK. The parity integrity mechanism (i.e. CRC and parity) is provided by the frame of the SECTOR SELECT command, see section 4.

Table 5: SECTOR SELECT Command

	<i>Name</i>	<i>Code (1 byte)</i>	<i>Parameter (1 byte)</i>	<i>Data</i>	<i>Integrity mechanism</i>
Command Packet 1	SECTOR SELECT	C2h	FFh	-	CRC, Parity
Response	ACK	See section 5.1.4 (4 bits)	-	-	-
	NACK	See section 5.1.4 (4 bits)	-	-	-
Command Packet 2	SECTOR SELECT	-	SecNo: 00h-FEh, FFh RFU	-	CRC, Parity
Response	Passive ACK (no answer)	-	-	-	-
	Any response within 1 ms SHALL be interpreted as NACK	-	-	-	-

The NFC Forum device timeout value for the SECTOR SELECT Command Packet 1 is 1msec. I.e. the NFC Forum device after sending a SECTOR SELECT Command Packet 1 SHALL wait up to 1msec for incoming responses from the Type 2 tag platform before triggering a timeout.

The NFC Forum device after sending the SECTOR SELECT Command Packet 2 SHALL wait 1ms. If any response is received during this time it is interpreted as a NACK.

**Figure 7: Timing Overview SECTOR SELECT Command**

The NFC Forum device SHALL use the SECTOR SELECT command to address physical memory bigger than 1 KB (>1024 bytes). Using SECTOR SELECT it is possible to select a specific memory sector. The size of each sector is 1Kbyte.

SECTOR SELECT command is divided into two packets called Command Packet 1 and Command Packet 2 that the NFC Forum device SHALL be able to send in two different times. The Command Packet 1 contains the 1 byte command code (equal to C2h), the 1 byte parameter (equal to FFh), and the 2 bytes CRC. The NFC Forum device SHALL send the SECTOR SELECT Command Packet 1 as first. A Type 2 tag platform with the memory smaller than 1KB answers with a NACK to a SECTOR SELECT Command Packet 1. Type 2 tag platform with the memory bigger than 1KB answers with an ACK to a SECTOR SELECT Command Packet 1. After receiving an ACK the NFC Forum device SHALL send the Command Packet 2. This is a four byte with CRC and parity packet.

The first byte in the Command Packet 2 is the chosen sector number (SecNo). The following 3 bytes are reserved for future use, and set to 00h. If the sector number is inside the addressable data memory no further packet is sent (passive ACK), and the sector is selected accordingly by the Type 2 tag platform. The NFC Forum device SHALL interpret any response within 1 ms as NACK. Any response is replied by the Type 2 tag platform, if the addressable data memory space is exceeded.

The chosen sector stays valid, until a protocol infringement occurs or a new sector is addressed. The default sector after power up (and after a protocol infringement) is the 00h one (called sector 0).

The SECTOR SELECT command can select up to 254 different sectors: the SecNo range is from 00h to FEh, the value FFh is reserved for future use (RFU).

5.1.4 ACK and NACK

The ACK and NACK are command responses of the Type 2 tag platform.

Table 6: ACK and NACK

	<i>Name</i>	<i>Code (4 bits)</i>	<i>Parameter</i>	<i>Data</i>	<i>Integrity mechanism</i>
Response	ACK	Ah	-	-	-
Response	NACK	5h or 1h	-	-	-

The command code is Ah for ACK and 5h or 1h for NACK. The frame of the ACK and NACK command is described in section 4.

No CRC and parity integrity mechanism is used.

6 NDEF Detection and Access

This section describes how the NFC Forum device stores and accesses the NFC Forum data in the Type 2 tag platform.

6.1 NDEF Management

The NFC Forum device reads the Capability Container (CC) to detect and access the NFC Forum defined data inside the Type 2 tag platform. The CC contains NFC Forum management data.

The CC is stored in the block 3 of the static or dynamic memory structure (see chapter 2). The CC bytes can be written using the WRITE command (see section 5.1.2). The 4 data bytes of the WRITE command and the current contents of the 4 CC bytes are bit-wise “or-ed” and the result is the new contents of the CC bytes. This process is irreversible. If a bit is set to 1b, it cannot be changed back to 0b again.

The NFC Forum device SHALL NOT use the CC to store any application related data.

The NFC Forum device SHALL code the CC bytes of block 3 in the following way:

1. Byte 0 is equal to E1h (magic number) to indicate that NFC Forum defined data is stored in the data area (see chapter 2).
2. Byte 1 is the version number of this document supported by the Type 2 tag platform (see also section 6.1.1). The most significant nibble (i.e. the 4 most significant bits) indicates the major version number, and the least significant nibble (the 4 least significant bits) indicates the minor version number. The version number of this specification has major version number equal to 1h and minor version number equal to 0h i.e. the version is v1.0 and Byte 1 is equal to 10h.
3. Byte 2 indicates the memory size of the data area of the Type 2 tag platform. The value of byte 2 multiplied by 8 is equal to the data area size measured in bytes. For examples:
 - a. 48 bytes are indicated by byte 2 value equal to 06h,
 - b. 128 bytes are indicated by byte 2 value equal to 10h,
 - c. 2040 bytes are indicated by byte 2 value equal to FFh.
4. Byte 3 indicates the read and write access capability of the data area and CC area of the Type 2 tag platform.
 - d. The most significant nibble (the 4 most significant bits) indicates the read access condition:
 - i. The value 0h indicates read access granted without any security.
 - ii. The values from 1h to 7h and Fh are reserved for future use.
 - iii. The values from 8h to Eh are proprietary.
 - e. The least significant nibble (the 4 least significant bits) indicates the write access condition:
 - i. The value 0h indicates write access granted without any security.
 - ii. The values from 1h to 7h are reserved for future use.
 - iii. The values from 8h to Eh are proprietary.
 - iv. The value Fh indicates no write access granted at all.

Table 7 shows an example of coding of the CC bytes. The example is related to a Type 2 tag platform:

- With NFC Forum defined data (byte 0 = E1h),
- Supporting the version 1.0 (major number 1h, minor number 0h) of the mapping document (byte 1 = 10h),
- With 128 bytes of the data area size (byte 2 = 10h), and
- With read and write access granted without any security (byte 3 = 00h).

Table 7: Example of coding of the CC bytes of block 3

Byte 0	Byte 1	Byte 2	Byte 3
E1h	10h	10h	00h

6.1.1 Version Treating

The Byte 1 of the CC contains the version of the applied mapping document to the Type 2 tag platform. The mapping document version is indicated with two numbers: major number version and minor version number.

The handling of the different mapping document version numbers applied to the Type 2 tag platform (called T2VNo) and the one implemented in the NFC Forum device (called NFCDevVNo) is explained in the 4 cases of Table 8.

Table 8: Handling of the mapping document version numbers

No	Version Number Case	Handling
1	Major NFCDevVNo is equal to major T2VNo, and minor NFCDevVNo is bigger than or equal to minor T2VNo	The NFC Forum device SHALL access the Type 2 tag platform and SHALL use all features of the applied mapping document to this Type 2 tag platform.
2	If major NFCDevVNo is equal to major T2VNo, and minor NFCDevVNo is lower than minor T2VNo	Possibly not all features of the Type 2 tag platform can be accessed. The NFC Forum device SHALL use all its features and SHALL access this Type 2 tag platform.
3	If major NFCDevVNo is smaller than major T2VNo	Incompatible data format. The NFC Forum device cannot understand the Type 2 tag platform data. The NFC Forum device SHALL reject this Type 2 tag platform.
4	If major NFCDevVNo is bigger than major T2VNo	The NFC Forum device might implement the support for previous versions of this specification in addition to its main version. In case the NFC Forum device has the support from previous version, it SHALL access the Type 2 tag platform. On the contrary, in case the NFC Forum device has not the support from previous version, it SHALL reject the Type 2 tag platform.

NOTE Future versions of this specification have to define the allowed actions to an NFC Forum Tag with a version number lower than the version number of the NFC Forum device (e.g. whether it is allowed to upgrade the tag to the new version).

6.2 NDEF Storage

The data format of the NDEF message is defined in [NDEF]. The NDEF message is stored inside the value field of the NDEF Message TLV (see section 2.3.3) in the data area of the Type 2 tag platform (see chapter 2). In the following section the NDEF Message TLV is always the NDEF Message TLV detected by the NDEF detection procedure (see section 6.4.1).

6.3 Life Cycle

An NFC Forum device MAY detect a Type 2 tag platform in different states. The state is reflected by the content of the Type 2 tag platform. Every state has its own valid operations.

The state transitions are only relevant for NFC Forum devices, which are capable of writing Type 2 tag platform.

The Type 2 tag platform can be in one of the following states: INITIALISED, READ/WRITE or READ-ONLY.

If the Type 2 tag platform is not in a valid state according to the specification of the NFC Forum, the NFC Forum device SHALL ignore the Type 2 tag platform and its data. The reasons might be:

- Misconfigured CC area.
- Not allowed NDEF Read Procedure (see section 6.4.2), if Type 2 tag is in READ/WRITE or READ-ONLY state.
- Not allowed NDEF Write Procedure (see section 6.4.3), if Type 2 tag is in INITIALISED or READ/WRITE state.
- Mismatch between overall TLV blocks length and actual length of the data area.
- Invalid TLV block.

6.3.1 INITIALISED State

In this state the NFC Forum device MAY modify the content of the NFC Forum defined data i.e. NDEF Message TLV in the Type 2 tag platform.

The NFC Forum device SHALL detect a Type 2 tag platform in INITIALISED state when:

1. The CC area is set as described in section 6.1 with byte 3 equal to 00h (read/write access granted),
2. The data area contains an NDEF Message TLV, and
3. The length field of the NDEF Message TLV is equal to 00h.

6.3.2 READ/WRITE State

In this state the NFC Forum device MAY modify the content of the NFC Forum defined data i.e. NDEF Message TLV in the Type 2 tag platform.

The NFC Forum device SHALL detect a Type 2 tag platform in READ/WRITE state when:

1. The CC area is set as described in section 6.1 with byte 3 equal to 00h (read/write access granted),
2. The data area contains an NDEF Message TLV, and
3. The length field of the NDEF Message TLV is different from zero and equal to the actual length of the NDEF message in the value field.

6.3.3 READ-ONLY State

In this state the CC and the whole data area are set to read-only.

The NFC Forum device SHALL detect a Type 2 tag platform in READ-ONLY state when:

1. The CC area is set as described in section 6.1 with byte 3 equal to 0Fh (only read access granted),
2. The data area contains an NDEF Message TLV, and
3. The length field of the NDEF Message TLV SHALL be different from zero and equal to the actual length of the NDEF message in the value field.

NOTE To detect the READ-ONLY state the lock bits are not checked.

6.4 Command Sequence Description

In this section several procedures are described to manage NFC Forum defined data e.g. NDEF Message TLV that contains an NDEF message. The different state changes or transitions between the states of the Type 2 tag platform are shown in detail.

6.4.1 NDEF Detection Procedure

The NFC Forum device SHALL use the NDEF detection procedure to detect the presence of an NDEF message (see [NDEF]) inside a Type 2 tag platform. The NDEF Message TLV that is found by the NDEF detection procedure is also called mandatory NDEF Message TLV or first NDEF Message TLV.

The detection procedure is based on the control of byte 0 and 1 of the CC, and the presence of an NDEF Message TLV that MAY contain an NDEF message.

The NDEF detection procedure is the following:

1. Read the CC (block 3) using the READ command specified in section 5.1.1.
2. If byte 0 is equal to E1h and byte 1 describes the right version number (see section 6.1.1)) and the most significant nibble of byte 3 in block 3 is equal to 0h then go to item 3. Otherwise no NDEF data is detected in the Type 2 tag platform.
3. Read sequentially the data area using the READ command specified in section 5.1.1 starting from block 4 and search for NDEF Message TLV. Stop the searching as soon as a first NDEF Message TLV is found. If no NDEF message TLV is detected in the Type 2 tag platform, the tag is not in a valid state.

4. If the NDEF Message TLV is found:
 - a. If the length field is different from zero, the NDEF message (see [NDEF]) is detected in the Type 2 tag platform, or
 - b. If the length field is equal to zero no NDEF Message is detected. The tag might be in INITIALISED state.

If the data to be read exceed one or more sectors, the NFC Forum device SHALL use the SECTOR SELECT command, see section 5.1.3.

NOTE The NDEF detection procedure does not relate to a valid NDEF message (see [NDEF]). It reads the length of the store NDEF data and does not parse the NDEF data itself.

6.4.2 NDEF Read Procedure

The NDEF read procedure makes use of the READ command. The NFC Forum device SHALL use the NDEF read procedure to read the NDEF message after having detected the NDEF message using the NDEF detection procedure (see section 6.4.1).

If the NDEF detection procedure does not detect the presence of the NDEF message inside the value field of the first NDEF Message TLV, the NFC Forum device SHALL NOT use the NDEF read procedure.

The NDEF read procedure uses one or more READ commands (see section 5.1.1) to retrieve the whole NDEF message from the NDEF Message TLV. The length of the NDEF message is provided from the length field of the NDEF Message TLV (see section 2.3.3).

If the data to be read exceed one or more sectors, the NFC Forum device SHALL use the SECTOR SELECT command (see section 5.1.3).

6.4.3 NDEF Write Procedure

The NFC Forum device SHALL use the NDEF write procedure to write NFC Forum defined data i.e. the NDEF message inside the first NDEF Message TLV of the Type 2 tag platform.

The NDEF write procedure uses the READ and WRITE commands (see section 5.1.1 and 5.1.2), and the NDEF detection procedure (see section 6.4.1).

The NFC Forum device SHALL only write the NDEF message into Type 2 tag platform in INITIALISED or READ/WRITE state.

During the NDEF write procedure the NFC Forum device SHALL ignore and jump over reserved memory areas or dynamic lock bit areas indicated by Memory Control TLVs or Lock Control TLVs.

The NDEF write procedure is the following:

1. Check if the Type 2 tag platform is in INITIALISED or READ/WRITE state, and use the NDEF detection procedure.

2. If the Type 2 tag platform is in INITIALISED or READ/WRITE state, the first NDEF Message TLV is found (with or without NDEF message in it, see item a and b in section 6.4.1), and the available memory size for the NDEF Message TLV is big enough to contain the NDEF message, the operations below are allowed to be done in the following order using one or more WRITE commands (see section 5.1.2):
 - a. The length field of the found NDEF Message TLV is set to one byte long and the value of the length field is set to 00h,
 - b. The new NDEF message is written in the memory area starting from: the 2nd byte after the tag field of the found NDEF Message TLV if the new NDEF Message length is less than 255 bytes, or 4th bytes after the tag field of the found NDEF Message TLV if the new NDEF Message length is bigger than 254 bytes, and
 - c. The 1 or 3 bytes of the length field of the found NDEF Message TLV is updated with the length of the new NDEF message.

Otherwise if no NDEF Message TLV is found or the Type 2 tag platform is not in INITIALISED or READ/WRITE state or not enough memory space is available in the Type 2 tag platform, the NDEF message is not written in the Type 2 tag platform.
3. The Terminator TLV is written in the next byte after the first NDEF Message TLV using the WRITE command (see section 5.1.2), if the NDEF Message TLV block does not end at the last byte of the available data area. The Terminator TLV is not written, if the NDEF Message TLV block ends at the last byte of the available data area.

Concerning the operation item b, the writing of the value field of the found NDEF Message TLV leaves 1 or 3 bytes for the length field (see section 2.3) that are needed by the next operation item c to store the length of the NDEF message.

For the WRITE command the reading of not completely updated blocks is needed first, see section 5.1.2. If the data to be read or written exceed one or more sectors, the NFC Forum device SHALL use the SECTOR SELECT command, see section 5.1.3.

NOTE The NDEF write procedure over-writes the first NDEF Message TLV, and makes unusable the TLV blocks that were stored after it.

6.4.4 State Changes

This section describes the possible state changes (also called transitions) performed by the NFC Forum device. Figure 8 shows the states and the transitions between them. The transitions are valid for static and dynamic memory structure. Below the possible transitions used by the NFC Forum device for the Type 2 tag platform are listed:

- Transition from INITIALISED to READ/WRITE,
- Transition from INITIALISED to READ-ONLY, and
- Transition from READ/WRITE to READ-ONLY.

The NFC Forum device SHALL be able to perform the 3 transitions for static and dynamic memory structure.

NOTE A Type 2 tag platform might be issued in any valid state. So, a Type 2 tag platform might be issued in INITIALISED state, READ/WRITE state or even in READ-ONLY state having a predefined NDEF message stored on it.

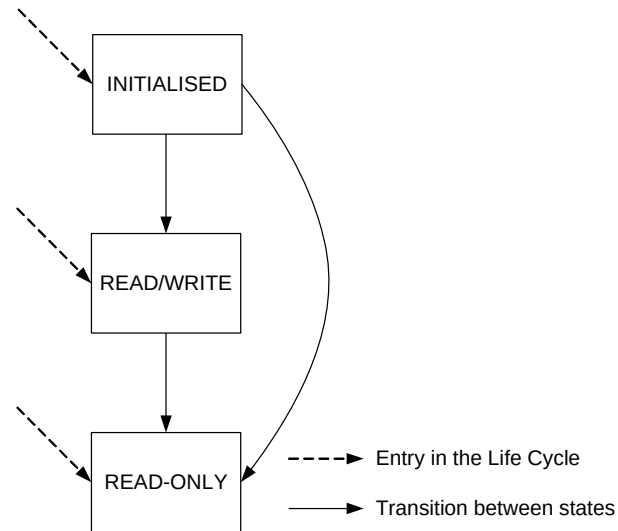


Figure 8: Life Cycle with State Changes (transitions)

6.4.4.1 Transitions from INITIALISED to READ/WRITE

To perform the transition from INITIALISED to READ/WRITE, the NFC Forum device **SHALL** use the NDEF write procedure (see section 6.4.3) to replace the empty NDEF Message TLV with a non empty NDEF Message TLV (length field different from zero).

The NFC Forum device **MAY** perform the transition only if the Type 2 tag platform is in INITIALISED state, otherwise it **SHALL NOT** perform it.

The transition from READ/WRITE to INITIALISED is not defined. To delete or remove the NDEF message in the NDEF Message TLV, the NFC Forum device **MAY** write an empty NDEF message (see Appendix A).

6.4.4.2 Transitions from READ/WRITE to READ-ONLY

To perform the transition from READ/WRITE to READ-ONLY the NFC Forum device **SHALL** set the Byte 3 of the CC to 0Fh, and all lock bits to 1b (including static and possible dynamic lock bits). To set bits and bytes, the NFC Forum device **MAY** use one or more WRITE commands.

The NFC Forum device **MAY** perform the transition only if the Type 2 tag platform is in READ/WRITE state, otherwise it **SHALL NOT** perform it.

If the Type 2 tag platform has a dynamic memory structure indicated by the CC byte 2 value bigger than 06h (see section 6.1), the NFC Forum device **SHALL** do the following operations to set to 1b the dynamic lock bits:

1. Check if any Lock Control TLV(s) is present in the data area.
2. If one or more Lock Control TLVs are present, set to 1b the dynamic lock bits identified by these Lock Control TLVs.
3. If no Lock Control TLV is present, set to 1b the dynamic lock bits identified by the default setting of the dynamic lock bits (see section 2.2.2).

For the WRITE command the reading of not completely updated blocks **SHALL** be done by the NFC Forum device first (see section 5.1.2). If the data to be read or written exceed one or more sectors, the NFC Forum device **SHALL** use the SECTOR SELECT command (see section 5.1.3).

6.4.4.3 Transition from INITIALISED to READ-ONLY

To perform the transition from INITIALISED to READ-ONLY, the NFC Forum device SHALL execute in the following order:

- The transition from INITIALISED to READ/WRITE (see section 6.4.4.1), and
- The transition from READ/WRITE to READ-ONLY (see section 6.4.4.2).

The NFC Forum device MAY perform the transition only if the Type 2 tag platform is in INITIALISED state, otherwise it SHALL NOT perform it.

A. Empty NDEF Message

An empty NDEF message (see [NDEF]) is defined as an NDEF message composed of one NDEF record. The NDEF record uses the NDEF short-record layout (SR=1b) with: Type Name Format (TNF) field value equal to 00h (empty, TYPE_LENGTH=00h, PAYLOAD_LENGTH=00h), no ID_LENGTH field (IL=0b), MB=1b, ME=1b, CF=0b. The empty NDEF record (i.e. the empty NDEF message) is composed of 3 bytes and it is equal to D00000h.

B. Memory Structure Examples

This appendix shows two examples of memory structure: the first one for the static memory structure and the second one for the dynamic memory structure.

B.1 Example of Static Memory Structure

This section describes an example of static memory structure with 16 blocks. The example is shown in Figure 9.

Byte Number	0	1	2	3	Block
UID / Internal	UID0	UID1	UID2	Internal0	0
Serial Number	UID3	UID4	UID5	UID6	1
Internal / Lock	Internal1	Internal2	Lock0 = 00h	Lock1 = 00h	2
CC	CC0 = E1h	CC1 = 10h	CC2 = 06h	CC3 = 00h	3
Data	NDEFMessage TLV0 = 03h	NDEFMessage TLV1 = 00h	TerminatorTLV0 = FEh	Data3	4
Data	Data4	Data5	Data6	Data7	5
Data	Data8	Data9	Data10	Data11	6
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	..	Data46	Data47	15

Figure 9: Example of Static Memory Structure.

The Type 2 tag platform is in INITIALISED state, the overall memory is set as following:

- All lock bits are set to 0b: Lock0 = 00h, and Lock1 = 00h
- The CC are is set following section 6.1:
 - CC0 = E1h to indicate that NDEF data is present inside the tag.
 - CC1 = 10h to indicate to support the version 1.0 (major number 1h, minor number 0h) of the mapping document i.e. the version of this specification,
 - CC2 = 06h to indicate 48 bytes of memory size assigned to the data area, dynamic lock bytes, and the reserved bytes,
 - CC3 = 00h to indicated read and write access granted without any security.

- The data area contains 2 TLV blocks in the following order:
 - NDEF Message TLV: empty TLV block
 - T = 03h
 - L = 00h
 - V = - (not present)
 - Terminator TLV:
 - T = FEh
 - L = - (not present)
 - V = - (not present)

B.2 Example of Dynamic Memory Structure

This section describes an example of dynamic memory structure with 32 blocks (k = 31), and 4 blocks of reserved data at the end of the physical memory. The example is shown in Figure 10.

Byte Number	0	1	2	3	Block
UID / Internal	UID0	UID1	UID2	Internal0	0
Serial Number	UID3	UID4	UID5	UID6	1
Internal / Lock	Internal1	Internal2	Lock0 = 00h	Lock1 = 00h	2
CC	CC0 = E1h	CC1 = 10h	CC2 = 0Ch	CC3 = 00h	3
Lock Control TLV	LockControlTLV0 = 01h	LockControlTLV1 = 03h	LockControlTLV2 = E0h	LockControlTLV3 = 06h	4
Lock Control TLV / Memory Control TLV	LockControlTLV4 = 33h	MemoryControlTLV0 = 02h	MemoryControlTLV1 = 03h	MemoryControlTLV2 = E1h	5
Memory Control TLV / NDEF Message TLV	MemoryControlTLV3 = 0Fh	MemoryControlTLV4 = 30h	NDEFMessageTLV0 = 03h	NDEFMessageTLV1 = 00h	6
Terminator TLV / Data	TerminatorTLV0 = FEh	Data13	Data14	..	7
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	Data93	Data94	Data95	27
Lock / Reserved	Lock2 = 00h	Reserved0	Reserved1	Reserved2	28
Reserved	Reserved3	Reserved4	Reserved5	Reserved6	29
Reserved	Reserved7	Reserved8	Reserved9	Reserved10	30
Reserved	Reserved11	Reserved12	Reserved13	Reserved14	31

Figure 10: Example of Dynamic Memory Structure.

The tag is in INITIALISED state, the main memory area is set as following:

- All lock bits are set to 0b: Lock0 = Lock1 = Lock2 = 00h
- The CC are is set following section 6.1:
 - CC0 = E1h to indicate that NDEF data is present inside the tag.
 - CC1 = 10h to indicate to support the version 1.0 (major number 1h, minor number 0h) of the mapping document,
 - CC2 = 0Ch to indicate 96 bytes of memory size assigned to the data area,
 - CC3 = 00h to indicated read and write access granted without any security.
- The data area contains 4 TLV blocks in the following order:
 - Lock Control TLV:
 - T = 01h
 - L = 03h
 - V = E00633h indicates that each lock bit locks 1 page, each page is 8bytes, and the lock area is 1 byte long at the byte address

$$\text{ByteAddr} = \text{PageAddr} * 2^{\text{BytesPerPage}} + \text{ByteOffset} = 14 * 2^3 + 0 = 112 \text{ where:}$$
 - Position = E0h contains PageAddr = Eh, and ByteOffset = 0h
 - Size = 06h
 - PageControl = 33h contains BytesPerPage = 3h ($2^3 = 8$ bytes), and BytesLockedPerLockBit = 3h ($2^3 = 8$ bytes).
 - Memory Control TLV:
 - T = 02h
 - L = 03h
 - V = E10F 30h indicates that the reserved area is 15 byte long at the byte address is

$$\text{ByteAddr} = \text{PageAddr} * 2^{\text{BytesPerPage}} + \text{ByteOffset} = 14 * 2^3 + 1 = 113 \text{ where:}$$
 - Position = E1h contains PageAddr = Eh, and ByteOffset = 1h
 - Size = 0Fh
 - PageControl = 30h contains BytesPerPage = 3h, and least significant nibble = 0h (RFU, ignored)
 - NDEF Message TLV: empty TLV block
 - T = 03h
 - L = 00h
 - V = - (not present)

- Terminator TLV:
 - T = FEh
 - L = - (not present)
 - V = - (not present)

The Lock Control TLV MAY be skipped using the default values for the dynamic lock area (see section 2.2.2). Also the Memory Control TLV can be skipped being after the data area and after the dynamic lock area (see Figure 10 block 28-31).

C. Examples of Command Flow

This chapter provides some examples of the command flow in order to show how a typical interaction can be performed. It is assumed that the Type 2 tag platform is in INITIALISED or in READ/WRITE state. This example does not cover any checks of the NDEF message.

The commands and the responses are written in hexadecimal form with a space between each byte (e.g. 30 F3 AB 9C) without the “h” character at the end. The left-most byte is the first byte sent, and the right-most byte is the last byte sent. Special acronyms like CRC0, CRC1, Data3, Data11... are written to indicate a group of data or bytes with a specific meaning indicated later on in the description of the command or of the response.

C.1 Static Memory Structure Examples

The two examples in the following two sections are based on static memory structure (see section 2.1).

C.2 Detection of NDEF Message

To detect the NDEF message the NDEF detection procedure is applied (see section 6.4.1). Two examples are given with a positive and a negative detection of NDEF message.

C.3 Positive Detection of NDEF Message

A READ command to read blocks 3 to 6 (BNo=3-6) of the static memory structure where the CC and NDEF Message TLV are located.

Command: 30 03 CRC0 CRC1

- The meanings of the bytes are the following ones:
 - 30h to indicate the READ command,
 - 03h to indicate to start to read from the block number (BNo),
 - CRC0 and CRC1 are the CRC bytes.

Response: E1 10 06 00 03 03 D0 00 00 FE Data6 ... Data11 CRC0 CRC1

- The meanings of the bytes are the following ones (note that only the first 4 bytes belong to block 3 and to the CC):
 - E1h = CC0 to indicate that NDEF data is present inside the tag.
 - 10h = CC1 to indicate to support the version 1.0 (major number 1h, minor number 0h) of the mapping document,
 - 06h = CC2 to indicate 48 bytes of memory size assigned to the data area,
 - 00h = CC3 to indicated read and write access granted without any security,
 - 03h to indicate that an NDEF Message TLV is present,
 - 03h indicate that the value field of the NDEF Message TLV is present (length equal to zero) and contains 3 bytes,
 - 0D0000h Empty NDEF message (see Appendix A),
 - FEh to indicate that the Terminator TLV is present,

- Data6...Data11 data area bytes containing not meaningful information. They are ignored during reading operations,
- CRC0 and CRC1 are the CRC bytes.

The NDEF message is detected inside the Type 2 tag platform because the L field of the NDEF Message TLV is different from 00h. The NDEF detection procedure does not parse the V field of the NDEF Message TLV, but it checks the L field if it is different from 00h.

C.4 Negative Detection of NDEF Message

This example does follow the example in Figure 9.

A READ command to read blocks 3 to 6 (BNo=3-6) of the static memory structure where the CC and NDEF Message TLV are located.

Command: 30 03 CRC0 CRC1

- The meanings of the bytes are the following ones:
 - 30h to indicate the READ command,
 - 03h to indicate to start to read from the block number (BNo),
 - CRC0 and CRC1 are the CRC bytes.

Response: E1 10 06 00 03 00 FE Data3 ... Data11 CRC0 CRC1

- The meanings of the bytes are the following ones (note that only the first 4 bytes belong to block 3 and to the CC):
 - E1h = CC0 to indicate that NDEF data is present inside the tag.
 - 10h = CC1 to indicate to support the version 1.0 (major number 1h, minor number 0h) of the mapping document,
 - 06h = CC2 to indicate 48 bytes of memory size assigned to the data area,
 - 00h = CC3 to indicated read and write access granted without any security,
 - 03h to indicate that an NDEF Message TLV is present,
 - 00h indicate that the value field of the NDEF Message TLV is not present (length equal to zero),
 - FEh to indicate that the Terminator TLV is present,
 - Data3...Data11 data area bytes containing not meaningful information. They are ignored during reading operations.
 - CRC0 and CRC1 are the CRC bytes.

The NDEF message is not detected inside the Type 2 tag platform because the L field of the NDEF Message TLV is equal to 00h (i.e. not NDEF message is present in the V field of the NDEF Message TLV).

C.5 Read of an NDEF message from the Data Area

Having done the NDEF detection procedure, the reading of the NDEF message is done using the NDEF read procedure (see section 6.4.2) starting from the first block where the V field of the NDEF Message TLV begins. In the example of section 6.4.4.3C.3 the READ command can start from block 4 (BNo=4).

C.6 Write of an NDEF message in the Data Area

To write an NDEF Message TLV in the data area the NDEF write procedure is used (see section 6.4.3). The Type 2 tag platform is supposed to be in INITIALISED state.

In this example the writing of the NDEF Message TLV starts from the first block of the data area block 4 (BNo = 4) where the first NDEF Message TLV was found (see Figure 9). The Terminator TLV is written in the byte after the NDEF Message TLV.

An empty NDEF message D00000h (see Appendix A) is written inside an NDEF Message TLV. It is allowed to write the NDEF message because a previous NDEF Message TLV is found, and the memory space is able to contain the NDEF message D00000h. Concerning the latter point the available data area size for the NDEF Message TLV is equal to 48 bytes because the beginning of the found NDEF Message TLV is at the first byte of the data area (Byte 0 of block 4), and the overall data area size is equal to 48 bytes (Byte 2 of the CC, CC2). 48 bytes are big enough to store the NDEF Message TLV equal to 5 bytes: 1 byte T field, 1 byte L field, and 3 bytes V field (NDEF message D00000h).

A READ command is needed at the beginning because the writing of the NDEF Message TLV requires 3 WRITE commands, that all of them partially write a block.

Command: 30 04 CRC0 CRC1

- The meanings of the bytes are the following ones:
 - 30h to indicate the READ command,
 - 04h to indicate to start to read from the block number 4 (BNo=4),
 - CRC0 and CRC1 are the CRC bytes.

Response: 03 00 FE Data3 ... Data15 CRC0 CRC1

- The meanings of the bytes are the following ones (note that only the first 4 bytes belong to block 4 and to the CC):
 - 03h to indicate that an NDEF Message TLV is present (T field),
 - 00h indicate that the value field of the NDEF Message TLV is not present (length equal to zero),
 - FEh to indicate that the Terminator TLV is present,
 - Data3...Data16 data area bytes containing not meaningful information. They are ignored during reading operations.
 - CRC0 and CRC1 are the CRC bytes.

The first WRITE command sets the length field (or L field) of the NDEF Message TLV (see section 6.4.3) to 00h. At the same time this command writes the first two bytes of the value field D000h.

Command: A2 04 03 00 00 00 CRC0 CRC1

- The meanings of the bytes are the following ones:
 - A2h to indicate the WRITE command,
 - 04h to indicate to write the block 4 (BNo = 4),
 - 03h is the T field of the NDEF Message TLV, byte value previously read to be not modified,
 - 00h is the L field of the NDEF Message TLV, this byte will be modified later on as described by the NDEF write procedure (see section 6.4.3),
 - D0h is the first byte of the V field of the NDEF Message TLV, byte that was meant to be changed by this WRITE command,
 - 00 is the second byte of the V field of the NDEF Message TLV, byte that was meant to be changed by this WRITE command,
 - CRC0 and CRC1 are the CRC bytes.

Response: Ah

- This is the 4 bits ACK packet.

The second WRITE command writes the last byte of the value field of the NDEF Message TLV (00h), and the Terminator TLV (FEh).

Command: A2 05 00 FE Data6 Data7 CRC0 CRC1

- The meanings of the bytes are the following ones:
 - A2h to indicate the WRITE command,
 - 05h to indicate to write the block 5 (BNo = 5),
 - 00h third byte of the V field of the NDEF Message TLV,
 - FEh is the T field of the Terminator TLV,
 - Data6 byte value previously read to be not modified,
 - Data7 byte value previously read to be not modified,
 - CRC0 and CRC1 are the CRC bytes.

Response 2: Ah

- This is the 4 bits ACK packet.

The third WRITE command sets the length field (or L field) of the NDEF Message TLV (see section 6.4.3) to 03h.

Command: A2 04 03 00 D0 00 CRC0 CRC1

- The meanings of the bytes are the following ones:
 - A2h to indicate the WRITE command,
 - 04h to indicate to write the block 4 (BNo = 4),
 - 03h is the T field of the NDEF Message TLV, byte value previously read to be not modified,

- 03h is the L field of the NDEF Message TLV, only byte that was meant to be changed by this WRITE command,
- D0h is the first byte of the V field of the NDEF Message TLV, byte to be not modified,
- 00 is the second byte of the V field of the NDEF Message TLV, byte to be not modified,
- CRC0 and CRC1 are the CRC bytes.

Response: Ah

- This is the 4 bits ACK packet.

C.7 Dynamic Memory Structure Examples

The two examples in the following two sections are based on dynamic memory structure (see section 2.2).

C.8 Detection of NDEF Message

To detect the NDEF message the NDEF detection procedure is applied (see section 6.4.1). Two examples are given with one positive and one negative detection of NDEF message.

C.9 Positive Detection of NDEF Message

A READ command is used to read blocks 3 to 6 (BNo=3-6) of the static memory structure where the CC and TLV blocks are located.

Command: 30 03 CRC0 CRC1

- The meanings of the bytes are the following ones:
 - 30h to indicate the READ command,
 - 03h to indicate to start to read from the block number (BNo),
 - CRC0 and CRC1 are the CRC bytes.

Response: E1 10 0C 00 01 03 C0 01 31 02 03 C1 0F 30 03 03 CRC0 CRC1

- The meanings of the bytes are the following ones (note that only the first 4 bytes belong to block 3 and to the CC):
 - E1h = CC0 to indicate that NDEF data is present inside the tag.
 - 10h = CC1 to indicate to support the version 1.0 (major number 1h, minor number 0h) of the mapping document,
 - 0Ch = CC2 to indicate 96 bytes of memory size assigned to the data area,
 - 00h = CC3 to indicated read and write access granted without any security,
 - 0103C00131h to indicate a Lock Control TLV,
 - 0203C10F30h to indicate a Memory Control TLV,
 - 03h to indicate that an NDEF Message TLV is present,
 - 03h indicate that the value field of the NDEF Message TLV is present (the length of the value field is equal to 3 bytes),

- CRC0 and CRC1 are the CRC bytes.

The NDEF message is detected inside the Type 2 tag platform because the L field of the NDEF Message TLV is different from 00h. The NDEF detection procedure does not parse the V field of the NDEF Message TLV, but it checks the L field if it is different from 00h.

C.10 Negative Detection of NDEF Message

This example does follow the example in Figure 10.

A READ command is used to read blocks 3 to 6 (BNo=3-6) of the static memory structure where the CC and TLV blocks are located.

Command: 30 03 CRC0 CRC1

- The meanings of the bytes are the following ones:
 - 30h to indicate the READ command,
 - 03h to indicate to start to read from the block number (BNo),
 - CRC0 and CRC1 are the CRC bytes.

Response: E1 10 0C 00 01 03 C0 01 31 02 03 C1 0F 30 03 00 CRC0 CRC1

- The meanings of the bytes are the following ones (note that only the first 4 bytes belong to block 3 and to the CC):
 - E1h = CC0 to indicate that NDEF data is present inside the tag.
 - 10h = CC1 to indicate to support the version 1.0 (major number 1h, minor number 0h) of the mapping document,
 - 0Ch = CC2 to indicate 96 bytes of memory size assigned to the data area,
 - 00h = CC3 to indicated read and write access granted without any security,
 - 0103C00131h to indicate a Lock Control TLV,
 - 0203C10F30h to indicate a Memory Control TLV,
 - 03h to indicate that an NDEF Message TLV is present,
 - 00h indicate that the value field of the NDEF Message TLV is not present (length equal to zero),
 - CRC0 and CRC1 are the CRC bytes.

The NDEF message is not detected inside the Type 2 tag platform because the L field of the NDEF Message TLV is equal to 00h (i.e. not NDEF message is present in the V field of the NDEF Message TLV).

C.11 Read of an NDEF message from the Data Area

The example in section 6.4.4.3C.9 that describes a positive detection of an NDEF Message, is used by this section to read the NDEF message.

Having done the NDEF detection procedure, the reading of the NDEF message is done using the NDEF read procedure (see section 6.4.2) starting from the first block where the V field of the NDEF Message TLV begins. In the example of section 6.4.4.3C.9 the READ command starts from block 7 (BNo=7). The read NDEF message might be 0D0000h (Empty NDEF message, see Appendix A). The detailed of the READ command are not shown.

C.12 Write of an NDEF message in the Data Area

This example follows the example in Figure 10.

To write an NDEF Message TLV in the data area the NDEF write procedure is used (see section 6.4.3). In the example below an empty NDEF message D00000h is written inside an NDEF Message TLV. The Type 2 tag platform is supposed to be in INITIALISED state. A Terminator TLV is inserted after the NDEF Message TLV. Looking Figure 10 the byte where the NDEF Message TLV is found is byte 2 of block 6. Being the WRITE command page-wise a previous READ command is requested to get the values of byte 0 and 1 of block 6. Afterwards two WRITE commands are sent to store the NDEF Message TLV and the Terminator TLV. Overall 3 commands are used.

It is allowed to write the NDEF message because a previous NDEF Message TLV is found, and the memory space is able to contain the NDEF message D00000h. Concerning the latter point the available data area size for the NDEF Message TLV is equal to 86 bytes because: the overall data area size is equal to 96 bytes (Byte 2 of the CC, CC2) and the beginning of the found NDEF Message TLV is at the 11th byte of the data area (Byte 2 of block 6, the first 10 bytes of the data area are occupied by Lock Control TLV and Memory Control TLV). 86 bytes are big enough to store the NDEF Message TLV equal to 5 bytes: 1 byte T field, 1 byte L field, and 3 bytes V field i.e. NDEF message D00000h.

A READ command is needed at the beginning because the writing of the NDEF Message TLV requires WRITE commands that partially write a block.

Command: 30 06 CRC0 CRC1

- The meanings of the bytes are the following ones:
 - 30h to indicate the READ command,
 - 06h to indicate to start to read from the block number (BNo) 6,
 - CRC0 and CRC1 are the CRC bytes.

Response: 0F 30 03 00 FE Data13...Data23 CRC0 CRC1

- The meanings of the bytes are the following ones (note that only the first 4 bytes belong to block 3 and to the CC):
 - 0Fh byte 0 of block 6,
 - 30h byte 1 of block 6
 - 03h to indicate that an NDEF Message TLV is present (T field),
 - 00h indicate that the value field of the NDEF Message TLV is not present (length equal to zero),
 - FEh to indicate that the Terminator TLV is present,
 - Data13...Data23 data area bytes containing not meaningful information. They are ignored during reading operations.
 - CRC0 and CRC1 are the CRC bytes.

The first WRITE command sets the length field (or L field) of the NDEF Message TLV (see section 6.4.3) to 00h.

Command: A2 06 0F 30 03 00 CRC0 CRC1

- The meanings of the bytes are:
 - A2h to indicate the WRITE command,
 - 06h to indicate to write the block 6 (BNo = 6),
 - 0Fh byte 0 of block 6, this byte was read before and is not changed by the WRITE command,
 - 30h byte 1 of block 6, this byte was read before and is not changed by the WRITE command,
 - 03h is the T field of the NDEF Message TLV, this byte was read before and is not changed by the WRITE command,
 - 00h is the L field of the NDEF Message TLV,
 - CRC0 and CRC1 are the CRC bytes.

Response: Ah

- This is the 4 bits ACK packet.

The second WRITE command writes the value field of the NDEF Message TLV (i.e. D00000h), and the 1 byte Terminator TLV (i.e. FEh).

Command: A2 07 0D 00 00 FE CRC0 CRC1

- The meanings of the bytes are the following ones:
 - A2h to indicate the WRITE command,
 - 07h to indicate to write the next block 7 (BNo = 7),
 - D00000h is the V field of the NDEF Message TLV,
 - FEh is the T field of the Terminator TLV,
 - CRC0 and CRC1 are the CRC bytes.

Response: Ah

- This is the 4 bits ACK packet.

The third WRITE command sets the length field (or L field) of the NDEF Message TLV (see section 6.4.3) to 03h.

Command: A2 06 0F 30 03 03 CRC0 CRC1

- The meanings of the bytes are the following ones:
 - A2h to indicate the WRITE command,
 - 06h to indicate to write the block 6 (BNo = 6),
 - 0Fh byte 0 of block 6, read before and not changed,
 - 30h byte 1 of block 6, read before and not changed,
 - 03h is the T field of the NDEF Message TLV, read before and not changed,
 - 03h is the L field of the NDEF Message TLV
 - CRC0 and CRC1 are the CRC bytes.

Response: Ah

- This is the 4 bits ACK packet.

D. Revision History

The following table outlines the revision history of Type 2 Tag Operation.

Table 9: Revision History

Document Name	Revision and Release Date	Status	Change Notice	Supersedes
NFCForum-TS-Type-2-Tag_1.0	1.0, July 2007	Final	None	



Type 3 Tag Operation

Technical Specification

NFC Forum™

NFCForum-TS-Type-3-Tag_1.0

2007-08-16

RESTRICTIONS ON USE

This specification is copyright © 2005-2007 by the NFC Forum, and was made available pursuant to a license agreement entered into between the recipient (Licensee) and NFC Forum, Inc. (Licensor) and may be used only by Licensee, and in compliance with the terms of that license agreement (License). If you are not the Licensee, you are not authorized to make any use of this specification. However, you may obtain a copy at the following page of Licensor's Website: http://www.nfc-forum.org/resources/spec_license after entering into and agreeing to such license terms as Licensor is then requiring. On the date that this specification was downloaded by Licensee, those terms were as follows:

1. LICENSE GRANT.

Licensor hereby grants Licensee the right, without charge, to copy (for internal purposes only) and share the Specification with Licensee's members, employees and consultants (as appropriate). This license grant does not include the right to sublicense, modify or create derivative works based upon the Specification.

2. NO WARRANTIES.

THE SPECIFICATION IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY, COMPLETENESS AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL LICENSOR, ITS MEMBERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE SPECIFICATION.

3. THIRD PARTY RIGHTS.

Without limiting the generality of Section 2 above, LICENSOR ASSUMES NO RESPONSIBILITY TO COMPILE, CONFIRM, UPDATE OR MAKE PUBLIC ANY THIRD PARTY ASSERTIONS OF PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS THAT MIGHT NOW OR IN THE FUTURE BE INFRINGED BY AN IMPLEMENTATION OF THE SPECIFICATION IN ITS CURRENT, OR IN ANY FUTURE FORM. IF ANY SUCH RIGHTS ARE DESCRIBED ON THE SPECIFICATION, LICENSOR TAKES NO POSITION AS TO THE VALIDITY OR INVALIDITY OF SUCH ASSERTIONS, OR THAT ALL SUCH ASSERTIONS THAT HAVE OR MAY BE MADE ARE SO LISTED.

4. TERMINATION OF LICENSE.

In the event of a breach of this Agreement by Licensee or any of its employees or members, Licensor shall give Licensee written notice and an opportunity to cure. If the breach is not cured within thirty (30) days after written notice, or if the breach is of a nature that cannot be cured, then Licensor may immediately or thereafter terminate the licenses granted in this Agreement.

5. MISCELLANEOUS.

All notices required under this Agreement shall be in writing, and shall be deemed effective five days from deposit in the mails. Notices and correspondence to the NFC Forum address as it appears below. This Agreement shall be construed and interpreted under the internal laws of the United States and the Commonwealth of Massachusetts, without giving effect to its principles of conflict of law.

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

Contents

1	Introduction.....	1
1.1	Objectives.....	1
1.2	Purpose	1
1.3	Applicable Documents or References	1
1.4	Administration.....	1
1.5	Special Word Usage	1
1.6	Name and Logo Usage	2
1.7	Intellectual Property	2
1.8	Glossary.....	2
1.9	Convention and notations.....	4
1.9.1	Representation of numbers.....	4
2	Memory Structure and Management.....	5
2.1	Blocks.....	5
2.2	Services	5
2.3	System Information	5
2.3.1	Manufacturer ID Information.....	5
2.3.2	System Definition Information.....	6
2.3.3	Service Definition Information	7
3	RF Interface.....	8
4	Framing and Transmission handling.....	9
4.1	Frame structure.....	9
4.2	Communication protocol.....	9
4.3	Communication endpoints.....	9
5	Command Set	10
5.1	State diagram (informative).....	10
5.2	Polling Command.....	10
5.3	Check Command	10
5.3.1	Command	10
5.3.2	Response.....	11
5.4	Update Command.....	13
5.4.1	Command	13
5.4.2	Response.....	14
5.5	Block List	15
5.5.1	Block List Format.....	15
5.6	Status-flags	16
5.6.1	Status-flag1.....	16
5.6.2	Status-flag2.....	16
6	NDEF Detection and Access.....	17
6.1	NDEF management data	17
6.1.1	System Code.....	17
6.1.2	Attribute Information Block.....	17
6.1.3	Versioning	19
6.2	NDEF storage	20
6.3	Life cycle.....	21
6.4	Command sequence description.....	22
6.4.1	NDEF detection.....	22

6.4.2	Read NDEF message.....	23
6.4.3	Write NDEF message.....	24
6.4.4	State changes	25

Figures

Figure 1: Manufacturer ID.....	5
Figure 2: Maximum Response Time Parameter	6
Figure 3: Type 3 Tag state diagram.....	10
Figure 4: Block Data	13
Figure 5: Two Bytes Block List Element	15
Figure 6: Three Bytes Block List Element	15
Figure 7: Mapping of NDEF to a Type 3 Tag	17
Figure 8: Data Blocks in Service.....	18
Figure 9: Attribute Information Format	18
Figure 10: Relation between NDEF data and blocks storing it	21
Figure 11: Life cycle states.....	22
Figure 12: General sequence for detecting NDEF data.....	22
Figure 13: General Sequence for reading NDEF data	24
Figure 14: General Sequence for writing NDEF data	25

Tables

Table 1: Access Attributes.....	7
Table 2: Check Command Data Structure.....	11
Table 3: Check Response Data Structure	12
Table 4: Update Command Data Structure.....	13
Table 5: Update response Data Structure	14
Table 6: Error Code List.....	16
Table 7: Handling of the mapping document version numbers.....	19
Table 8: NDEF Services	20
Table 9: Revision History.....	27

1 Introduction

This specification is part of the NFC Forum documentation about tag types that an NFC Forum device needs to support in reader/writer mode.

The Type 3 Tag implements a contactless interface that is compatible with the 212 kbps and 424 kbps modes of ISO/IEC 18092 (see [ISO/IEC 18092]).

1.1 Objectives

This specification documents how an NFC Forum Device can operate an NFC Forum Type 3 Tag. This is not a specification of the Type 3 Tag itself, but how an NFC Forum Device can make use of this tag type.

1.2 Purpose

The purpose of this specification is to document the requirements and to specify, with a set of rules and guidelines, the NFC Forum Device operation and management of a Type 3 Tag.

This specification also defines the data mapping and how the NFC Forum Device detects, reads, and writes NDEF data into the Type 3 tag platform in order to achieve and maintain interchangeability and interoperability.

1.3 Applicable Documents or References

[ISO/IEC 18092]	ISO/IEC 18092, Information Technology- Telecommunications and information exchange between systems- Near Field Communication - Interface and Protocol (NFCIP-1).
[NDEF]	“NFC Data Exchange Format (NDEF)” NFC Forum™, May 2006.
[RFC 2119]	S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Harvard University, March 1997.
[DIGPROT]	“NFC Digital Protocol Specification”, NFC Forum™, to be released
[ANINT]	“NFC Analog Interface Specification”, NFC Forum™, to be released

1.4 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.

1.5 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 Type 3 Tag Operation 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

Access Attribute

Indicate the access control mode like "read/write" or "read only"

Block

The smallest data unit that is applicable for writing and reading.

Block List

Set of Block List Elements used in the Check and Update commands.

Block List Element

Indicates the block number, the position of the Service within the Service Code List, and the access mode for one block

Block number

Indicates the logical position of a data block belonging to a specific Service. The NFC Forum Device uses this number to access the memory data.

IDm

Manufacturer ID. This is a 8 bytes number used for addressing a Type 3 Tags.

Little Endian

A method of recording or transmitting numerical data of more than 2 bytes, with the lowest byte placed at the beginning.

Ln

Length of the NDEF data in bytes

LSB

least significant byte

lsb

least significant bit

Manufacturer ID Information

Initial Type 3 Tag information, which consists of IDm and PMm.

MSB

most significant byte

msb

most significant bit

Nbc

Number of blocks used by the NDEF data stored on the Type 3 Tag.

Nbr

Number of blocks that can be read by the Check command at once.

Nbw

Number of blocks that can be written by the Update command at once.

NDEF

NFC Data Exchange Format

Nmaxb

Maximum number of blocks available for NDEF data on a Type 3 Tag

PMm

Manufacturer Parameter that is pre-configured by the Type 3 Tag manufacturer.

NFC Forum Device

NFC Forum compliant device. For this document the NFC Forum Device is always acting in Reader/Writer mode, which is similar to a Proximity Coupling Device (PCD) in ISO terminology.

RWflag

This flag shows whether the Type 3 Tag is “read only” or allows “read/write” access.

Service

Defines the method of accessing user blocks stored in the Type 3 Tag. A Type 3 Tag can contain more than one Service.

Service Code List

List of Service Codes that indicate which Services can be accessed

Service Definition Information

Specifies the access attributes, as well as the position and number of accessible user blocks, for a specific Service.

Status Flag

Indicates the Type 3 Tag's success/error conditions.

System Definition Information

Used for command/response sequence control between NFC Forum Device and Type 3 Tag.

It consists of Manufacturer ID Information, System Definition Information, and Service Definition Information.

Type 3 Tag

NFC Forum Type 3 compatible Tag or card or token including a contactless IC chip, which has build-in memory and memory access functions. The physical shape is not defined.

WriteFlag

Flag which indicates that the process of writing is currently ongoing or not.

1.9 Convention and notations

1.9.1 Representation of numbers

The following conventions and notations apply in this document unless otherwise stated.

- Binary numbers are represented by strings of digits 0 and 1 shown with the most significant bit (msb) left and the least significant bit (lsb) right, “b” is added at the end.
Example: 11110101b
- Hexadecimal numbers are represented using the numbers 0 - 9 and the characters A – F, an “h” is added at the end. The most significant byte (MSB) is shown on the left, the least significant byte (LSB) on the right.
Example: F5h
- Decimal numbers are represented as is (without any tailing character).
Example: 245

2 Memory Structure and Management

2.1 Blocks

The basic unit of information used in memory management is called a block. Each block has a fixed size of 16 bytes. The number of memory blocks available depends on the chip hardware. Memory blocks are not addressed directly but relative to the Service they belong to.

2.2 Services

Services are similar to files in a file system. Each Service has a number of memory blocks associated with it. Services can be addressed using their Service Code, which must be unique inside each Type 3 Tag.

2.3 System Information

Each Type 3 Tag contains management data, called System Information. The system information of a Type 3 Tag consists of the following parts:

- Manufacturer ID Information
- System Definition Information
- Service Definition Information

Manufacturer ID Information and System Definition Information are pre-assigned by the Type 3 Tag manufacturer.

2.3.1 Manufacturer ID Information

Manufacturer ID Information consists of the Manufacturer ID (IDm) and the Manufacturer Parameter (PMm). The Manufacturer ID Information cannot be deleted or re-written by users.

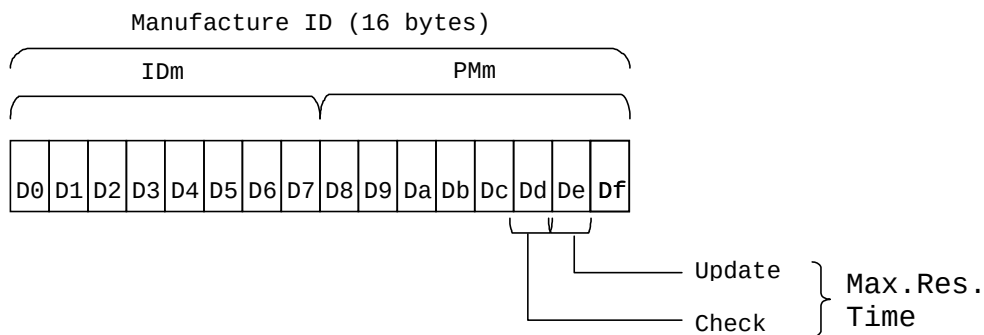


Figure 1: Manufacturer ID

2.3.1.1 Manufacturer ID (IDm)

The Manufacturer ID (IDm) is an 8 bytes number that is used by the NFC Forum device to address a Type 3 Tag.

2.3.1.2 Manufacturer Parameter (PMm)

The Manufacturer Parameter (PMm) contains the Maximum Response Time parameter.

The Maximum Response Time Parameter is used to inform the NFC Forum Device about the maximum time needed by the Type 3 Tag to execute Check and Update commands. It has a length of 6 bytes.

The NFC Forum Device SHALL interpret the 4th byte (Dd) to contain the maximum response time information for the Check command and the 5th byte (De) to contain the maximum response time information for the Update command.

The byte format for the 4th and 5th byte SHALL be interpreted as shown in the following diagram:

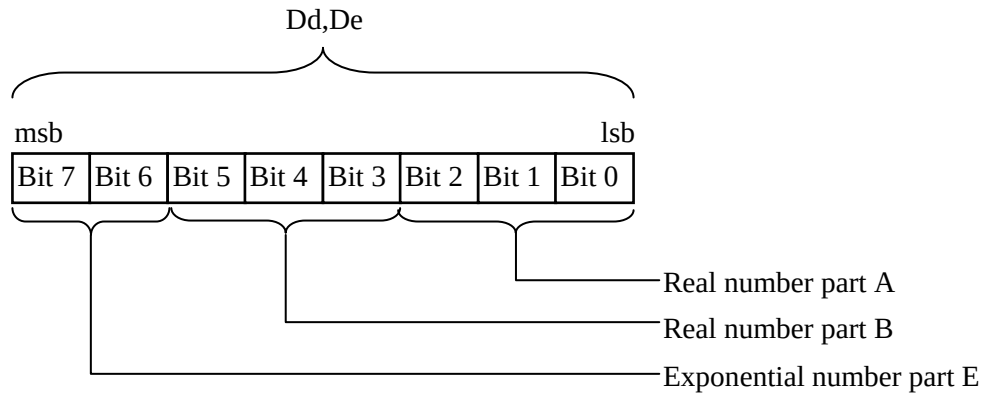


Figure 2: Maximum Response Time Parameter

The Maximum Response Time is defined by the chip manufacturer considering the maximum transaction time the Type 3 Tag needs to process the command. The response time, which is calculated using this parameter, is longer than the actual time needed for the processing of the corresponding command.

The following formula SHALL be used for the calculation of the maximum response time of the Check and Update commands:

$$\text{Maximum Response Time} = T \cdot [(B + 1) \cdot n + (A + 1)] \cdot 4^E$$

T : 0.302 ms

A : Real number part A

B : Real number part B

E : Exponential number part E

N : Number of blocks used in the Check/Update command

The Maximum Response Time of the Polling command is fixed.

2.3.2 System Definition Information

The System Definition Information consists of the System Code.

The System Code is a 2 bytes number. NFC Forum Devices can use the corresponding parameter in the Polling command to poll targets having a specific system code.

The System Code is coded in big endian order.

2.3.3 Service Definition Information

A Service Definition Information is present for each Service existing on a Type 3 Tag. It consists of the Service Code and the Number of Blocks for the Service.

The Service Code is uniquely identifying the Service on a Type 3 Tag. It has a length of 2 bytes. The format is little endian.

The Service Code consists of a Service Number and an Access Attribute. The Service Number has a length of 10 bits (the 10 most significant bits of the 2 bytes) and is unique for each Service of a Type 3 Tag. The Access Attribute has a length of 6 bits (the 6 least significant bytes) and specifies the permissions for accessing the associated memory blocks.

The Access Attribute SHALL be interpreted as described in the table below:

Table 1: Access Attributes

Value	Access Attribute	Comment
001001b	Read/Write	Data can be read or written.
001011b	Read only	Only reading data is possible.

The Number of Blocks is a 2 byte number specifying the number of memory blocks associated with this Service.

Each Service Definition Information usually references a number of memory blocks that are exclusively used by this Service. The only exceptions are Overlap Services. Overlap Services share the same memory blocks but have different Access Attributes (e.g. read only, read/write).

3 RF Interface

The RF interface of the NFC Forum Device is defined in [ANINT]. The NFC Forum Device SHALL comply with the RF interface as defined in the relevant clauses of [ANINT] for the 212/424kbps communication modes.

4 Framing and Transmission handling

4.1 Frame structure

The NFC Forum device SHALL send Check and Update commands and responses using the bit coding, character coding and frame coding defined in [DIGPROT] for NFC-F and Tag Type 3.

4.2 Communication protocol

The communication protocol SHALL be the half-duplex communication protocol defined in [DIGPROT] for the Tag Type 3.

The timeout value for the timeout mechanism of the NFC Forum device SHOULD be calculated using the maximum response time for each command, which can be derived from the information in the Manufacturer Parameter (see chapter 2.3.1.2).

4.3 Communication endpoints

The NFC Forum device and the Type 3 Tag are always the endpoints of the communication protocol described in chapter 4.1 as this protocol is a link layer protocol.

Nevertheless the exchange of commands and responses is a higher communication layer and happens between two entities capable of constructing and extracting the commands and responses.

The entity extracting the commands and constructing the responses is always part of the Type 3 Tag. The entity constructing the commands and extracting the responses may be part of the NFC Forum Device, but might also be part of a device connected via another network to the NFC Forum Device. In this case the NFC Forum Device forwards command Packet Data it received from an entity via another network to the Type 3 Tag, by encapsulating the Packet Data using the frame format described in [DIGPROT]. In a similar way, response Packet Data received from the Type 3 Tag must be send to the receiving entity by using the protocols of the other network. In this case, the NFC Forum Device does not need to interpret the data send to and received from the Type 3 Tag.

Remark: Application developers can offer a low level interface to send and receive arbitrary data to a Type 3 Tag. Such a low level interface enables a number of applications that require a remote entity to communicate with a Type 3 Tag.

5 Command Set

This chapter describes the command set of a Type 3 Tag. Additionally an informative state diagram of the Type 3 Tag is provided. These commands belong to the ‘proprietary protocol’ defined in [ISO/IEC 18092].

5.1 State diagram (informative)

This chapter shows the state diagram of a Type 3 Tag. This chapter is informative only.

A Type 3 Tag has only one state called “Mode 0”. In this state the Polling command, Check command and Update command can be received. None of these commands changes the state of the Type 3 Tag.

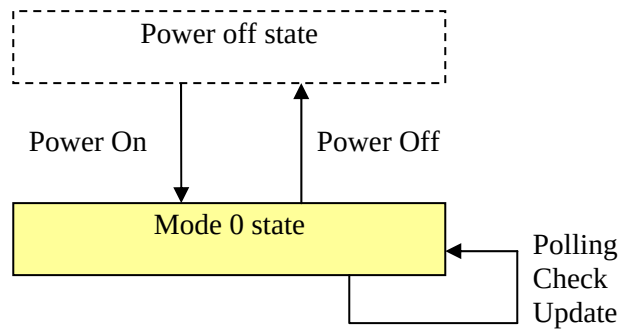


Figure 3: Type 3 Tag state diagram

5.2 Polling Command

The communication between an NFC Forum Device and a Type 3 Tag begins with the Polling command issued by the NFC Forum Device. The Polling command is used to recognize the Type 3 Tags in the field. It is also used for initialization and anti-collision.

The Polling command is fully specified in [DIGPROT] except for the following changes and additions:

- A type 3 Tag sends its IDm instead of the NFCID2 in the Polling response payload (see chapter 2.3.1.1 for information about the IDm).
- A Type 3 Tag sends its PMm instead of the Pad bytes of the Polling response payload (see chapter 2.3.1.2 for information about the PMm).

The terms Polling command and Polling response in this specification always refer to the corresponding specification given in this chapter and not to the specification in [ISO/IEC 18092] for Single Device Detection at 212 and 424 kbps.

5.3 Check Command

The Check command is used to read user data from the memory of a Type 3 Tag.

5.3.1 Command

The Command Code of the Check command SHALL be 06h.

The Check command SHALL include all data elements of the following table as command parameters:

Table 2: Check Command Data Structure

Data Elements	Bytes	Value	Comments
Manufacturer ID (IDm)	8		
Number of Services	1	n	It depends on the implementation of the Type 3 Tag how many Services can be read simultaneously. The minimum value is 1 and maximum is 16.
Service Code List	2n		
Number of Blocks	1	m	It depends on the implementation of the Type 3 Tag how many blocks can be read simultaneously. The minimum value is 1 and maximum is 12.
Block list	2m - 3m		See chapter 5.5

The Manufacturer ID (IDm) should be set to the value included in the Polling response of the targeted Type 3 tag.

The Block List SHALL be formatted as defined in chapter 5.5

The Number of Services in the Check command SHOULD be between 1 and the Maximum Number of Services (minimum value is 1 and maximum value is 16).

Every Service Code List Order value in the Block List Element SHOULD be smaller than Number of Services. The Maximum number of Service Code List Order is (n+1). The order number starts from 0 and is always smaller than Number of Services (n).

The Access Attributes of the Service Code that is specified by Block List Element SHOULD be 001001b (Read/Write) or 001011b (Read only).

The Number of Blocks in the Check command SHOULD be smaller than the Maximum Number of Blocks that is supported by the Type 3 Tag (minimum value is 1 and maximum value is 12).

All Services which are specified by the Service Code List Order in Block List Element SHOULD exist on the Type 3 Tag.

The Block number that is set in each Block List Element SHOULD be smaller than the Number of Blocks that is defined for the corresponding Service in the Service Definition Information (The maximum block number is (n+1) because the counting for this number starts from 0. This number is always smaller than Number of Blocks: n).

There MAY be Service Codes in Service Code List that are NOT referred by any of the Block List Elements in the Block List.

5.3.2 Response

NFC Forum Devices SHALL interpret responses with the response code of 07h as Check responses.

NFC Forum Devices SHALL interpret Check responses to be formatted as defined in the following table:

Table 3: Check Response Data Structure

Data Elements	Bytes	Value	Comments
Manufacturer ID (IDm)	8		
Status-flag1	1		00h: Success Others: Failure See Chapter 5.6
Status-flag2	1		See Chapter 5.6
Number of Blocks	1	m	This value is the same as Number of Blocks in the Check command. (If Status-flag1 = 00h)
Block Data	16m		(If Status-flag1 = 00h)

The NFC Forum Device SHOULD check that the IDm in the response is the same as the one sent in the Check command. The values of Status-flag1 and Status-flag2 SHALL be interpreted according to the definition given in Chapter 5.6.

The following data is included in the response if the Status-flag1 is equal to 00h:

- the Number of Blocks is the same value as in the Check command.
- Block Data is the data read from the memory blocks specified in the Check command. The order of Block Data is the same as the order of the Block List Elements in the Block List of the Check command.

Example:

Check Command:

Cmd. code	IDm	No. of Services	service code list	No. of Blocks	Block List		
06h		1	000Bh	3	Service order: 0 Block: 05h	Service order: 0 Block: 06h	Service order: 0 Block: 0Fh

Check Response:

Resp. code	IDm	Status-flag1	Status-flag2	No. of Blocks	Block Data		
07h		0	0	3			
					Service 000Bh, block 05h data	Service 000Bh, block 06h data	Service 000Bh, block 0Fh data

Figure 4: Block Data

The Type 3 Tag starts sending the Check response before the end of the Maximum Response Time, which is calculated using the Maximum Response Time Parameter for the Check command in the PMm. The Maximum Response Time period starts after the Type 3 Tag has received the Check command. The timeout of the NFC Forum Device to wait for the response from the Type 3 Tag SHALL be longer than the Maximum Response Time.

5.4 Update Command

The Update command is used to write user data to the memory blocks of a Type 3 Tag.

5.4.1 Command

The Command Code of the Update command SHALL be 08h.

The Update command SHALL include all data elements of the following table as command parameters:

Table 4: Update Command Data Structure

Data Elements	Bytes	Value	Comments
Manufacturer ID (IDm)	8		
Number of Services	1	n	It depends on the implementation of the Type 3 Tag how many Services can be written simultaneously. The minimum value is 1 and maximum is 16.
Service Code List	2n		
Number of Blocks	1	m	It depends on the implementation of the Type 3 Tag how many blocks can be written simultaneously. The minimum value is 1 and maximum is 8.
Block List	2m - 3m		See chapter 5.5
Block Data	16m		

The Manufacturer ID (IDm) should be set to the value included in the Polling response of the targeted Type 3 tag.

The Number of Services in the Update command SHOULD be between 1 and the Maximum Number of Services that is supported (minimum value is 1 and maximum value is 16).

Every Service Code List Order value in Block List Element SHOULD be smaller than Number of Services. The maximum number of Service Code List Order is (n+1). The order number starts from 0 and is always smaller than Number of Services (n).

All Services that are specified by the Service Code List Order in the Block List Element SHOULD exist in the Type 3 Tag memory.

The Access Attributes of the Service Code that is specified by Block List Element SHOULD be 001001b (Read/Write) or 001011b (Read only).

The Number of Blocks in the Update command SHOULD be smaller than the Maximum Number of Blocks that are supported. (Minimum value is 1 and maximum value is 8).

The block number that is set in each Block List Element SHOULD be smaller than the Number of Blocks that is defined for the corresponding Service in the Service Definition Information. The 'Maximum number of Block' value is (n+1) because it starts from 0. This number is always smaller than Number of Blocks (n).

There MAY be Service Codes in Service Code List that are NOT referred by any of the Block List Elements in the Block List.

5.4.2 Response

NFC Forum Devices SHALL interpret responses with the response code of 09h as Update responses.

NFC Forum Devices SHALL interpret Update responses to be formatted as defined in the following table:

Table 5: Update response Data Structure

Data Elements	Bytes	Value	Comments
Manufacturer ID (IDm)	8		
Status-flag1	1		00h: Success / Others: Failure see Chapter 5.6
Status-flag2	1		see Chapter 5.6

The NFC Forum Device SHOULD check that the IDm in the response is the same as the one sent in the Update command.

The values of Status-flag1 and Status-flag2 SHALL be interpreted according to the definition given in Chapter 5.6.

The Type 3 Tag starts sending the Update response before the end of the Maximum Response Time, which is calculated using the Maximum Response Time Parameter for the Update command in the PMm. The Maximum Response Time period starts after the Type 3 Tag has received the Update command. The timeout of the NFC Forum Device to wait for the response from the Type 3 Tag should be longer than the Maximum Response Time.

5.5 Block List

5.5.1 Block List Format

The Block List allows specifying multiple Blocks to be accessed in one command.

The Block List SHALL consist of one or more Block List Elements. Each Block List Element indicates a block number within the Service specified by its position in the Service Code List Order.

The following figure shows the two possible structures for Block List Elements:

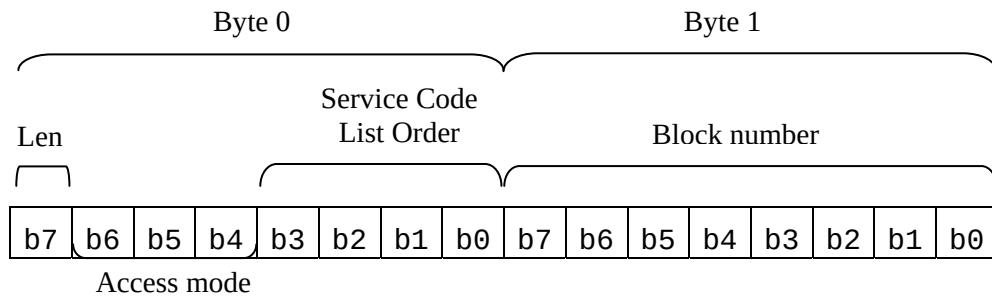


Figure 5: Two Bytes Block List Element

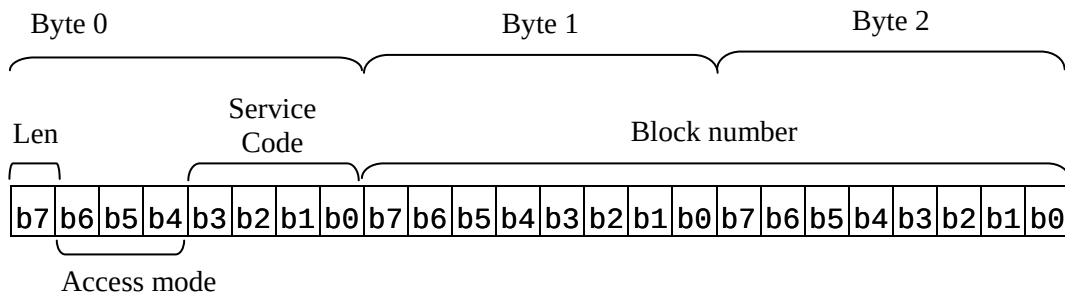


Figure 6: Three Bytes Block List Element

Block List Elements SHALL be either in a 2 byte or 3 byte format. Which format is used SHALL be specified by the Length bit which is the MSB of the first byte of each Block List Element.

The Length bit SHALL be set to 1b to indicate a 2 byte Block List Element. The Length bit SHALL be set to 0b to indicate a 3 byte Block List Element.

The bits 6, 5 and 4 of the first byte of each Block List Element are named Access Code and SHOULD be set to 000b.

The bits 3, 2, 1 and 0 of the first byte of each Block List Element are named Service Code List Order and indicate the Service the block referenced in the Block List Element belongs to. The Service Code List Order SHOULD contain the position of the corresponding Service in the Service Code List of the Check or Update commands. The first Service in the Service Code List has the number 0000b.

In case of a 2 byte Block List Element the 2nd byte SHOULD contain the block number.

In case of a 3 byte Block List Element the 2nd and 3rd byte SHOULD contain the block number in little endian format.

5.6 Status-flags

This chapter defines the allowed values of the status flags that indicate the Type 3 Tag's error conditions in a Check- or Update response.

The status is described using Status-flag1 (1 byte for indicating the error status and erroneous block) and Status-flag2 (1 byte for indicating more details of error information provided by Status Flag 1).

5.6.1 Status-flag1

The Status-flag1 is set to 00h in case the command execution was successful.

In case of an error, another value than 00h is set.

5.6.2 Status-flag2

The Status-flag2 is set to 00h in case the command execution was successful.

The values 70h and 71h indicate hardware errors of the Type 3 Tag. Any other values indicate processing errors.

Table 6: Error Code List

Error name	Explanation	Code
Memory Error	Cannot write to memory.	70
Excessive Writes	Memory has been written more than the maximum number of times. (This value depends on the hardware).	71

6 NDEF Detection and Access

This section describes how an NFC Forum device can store and access NDEF records on a Type 3 Tag using the commands described in chapter 5.

The procedures described in this chapter do only apply for storing NDEF on a Type 3 Tag. They do not describe how to use the commands described in chapter 5 for other use cases.

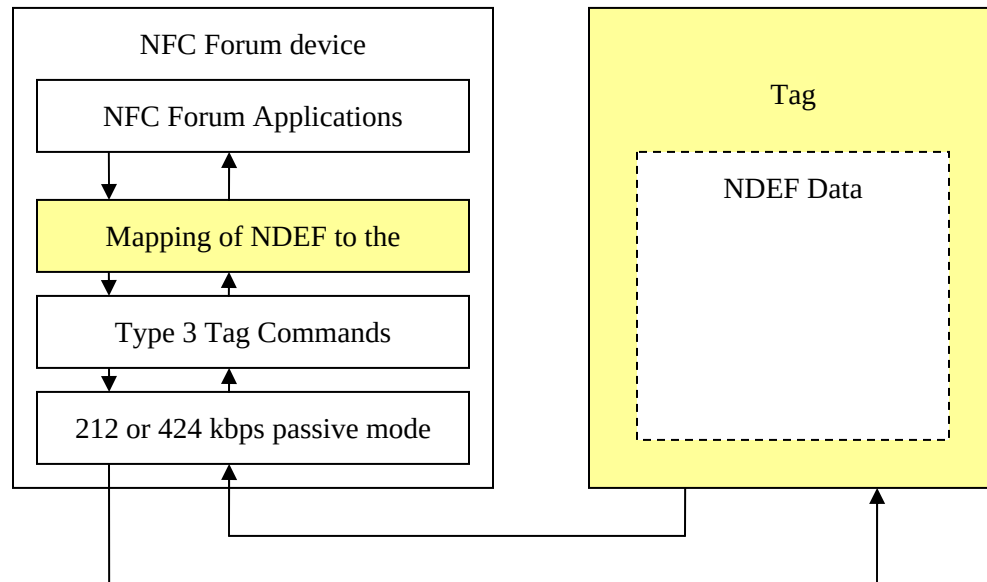


Figure 7: Mapping of NDEF to a Type 3 Tag

6.1 NDEF management data

6.1.1 System Code

The System Code 12FCh SHALL be used for NDEF enabled Type 3 Tags.

6.1.2 Attribute Information Block

The NDEF data SHALL be stored on the Type 3 Tag using the memory blocks assigned to the Service with the Service code 000Bh.

The Attribute Information Block SHALL be the 1st block of this Service. The data length is 16 bytes.

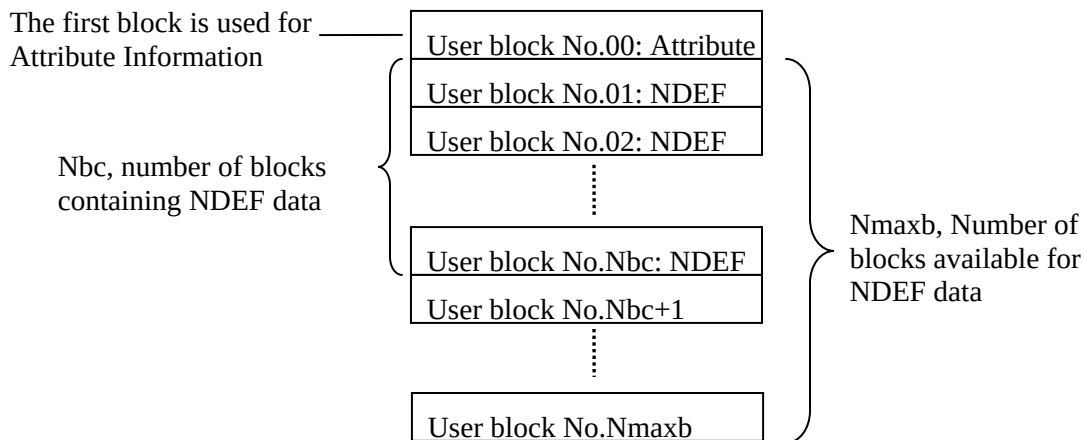


Figure 8: Data Blocks in Service

The following figure shows the structure of the Attribute Information Block:

User Block No.00															
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	Byte 15
Ver	Nbr	Nbw	Nmaxb		unused	unused	unused	unused	WriteF	RW Flag	Ln		Checksum		

Figure 9: Attribute Information Format

Byte 0 SHALL be the NDEF Mapping version. The version which corresponds to this specification is 1.0. The most significant nibble (the 4 most significant bits) SHALL indicate the major version number, and the least significant nibble (the 4 least significant bits) SHALL indicate the minor version number. The NFC Forum device SHALL not change the value of this field.

Byte 1 SHALL be Nbr. Nbr indicates the number of blocks that can be read using one Check command. The NFC Forum device SHALL not change the value of this field.

Byte 2 SHALL be Nbw. Nbw indicates the number of blocks that can be written using one Update command. The NFC Forum device SHALL not change the value of this field.

Byte 3 and Byte 4 SHALL be Nmaxb. Nmaxb indicates the maximum number of Blocks available for NDEF data. Byte 3 SHALL be the upper byte, Byte 4 SHALL be the lower byte of the maximum number of Blocks. The NFC Forum device SHALL not change the value of this field.

Byte 5 to Byte 8 are currently not used and SHALL not be changed by the NFC Forum device.

Byte 9 SHALL be WriteFlag. Allowed values for the WriteFlag SHALL be:

- 00h: OFF (Writing data finished)
- 0Fh: ON (Writing data in progress)

NFC Forum Devices SHOULD set this flag to ON before writing to the Type 3 Tag and set it back to OFF after writing.

Byte 10 SHALL be RWflag. Allowed values for the RWFlag SHALL be:

- 00h: Access Attribute: Read only.
- 01h: Access Attribute: Read/Write available.

An NFC Forum Device SHALL not try to write to a Type 3 Tag with this flag set to 00h, even if writing would be technically possible. Read-only Type 3 Tags always have this value set to 00h.

An NFC Forum Device SHALL not change the value of the RWflag.

Byte 11 to Byte 13 SHALL be Ln. Ln is the actual size of the stored NDEF data in bytes. Byte 11 SHALL be the upper byte, Byte 12 SHALL be middle byte and Byte 13 SHALL be the lower byte. The number of blocks containing NDEF data (Nbc) can be calculated by the following formula $Nbc = \lceil Ln/16 \rceil$.

An NFC Forum Device SHALL update the Ln field with a correct value each time NDEF data has been written.

Byte 14 and Byte 15 SHALL be a checksum calculated using the following formula:

$$Checksum = \text{Byte } 0 + \text{Byte } 1 + \dots + \text{Byte } 13$$

Byte 14 SHALL be the upper byte of the checksum, Byte 15 SHALL be the lower byte of the checksum. The NFC Forum device SHALL update the checksum with a correct value every time any of the values of Bytes 0 to 13 are changed.

6.1.3 Versioning

The Byte 0 of the Attribute Information Block contains the version of the applied mapping document to the NFC Forum Type 3 Tag. The mapping document version SHALL be indicated with two numbers: major version number and minor version number.

The handling of the different mapping document version numbers applied to the Type 3 Tag (called T3VNo) and the one implemented in the NFC Forum device (called NFCDevVNo) is explained in the 4 cases of Table 7.

Table 7: Handling of the mapping document version numbers

No	Version Number Case	Handling
1	Major NFCDevVNo is equal to major T3VNo, and minor NFCDevVNo is bigger than or equal to minor T3Vno	The NFC Forum Device SHALL access the Type 3 Tag and SHALL use all features of the applied mapping document to this Type 3 Tag.
2	If major NFCDevVNo is equal to major T3VNo, and minor NFCDevVNo is lower than minor T3Vno	Possibly not all features of the Type 3 Tag can be accessed. The NFC Forum Device SHALL use all its features and SHALL access this Type 3 Tag.
3	If major NFCDevVNo is smaller than major T3Vno	Incompatible data format. The NFC Forum Device cannot understand the Type 3 Tag data. The NFC Forum Device SHALL reject this Type 3 Tag.

No	Version Number Case	Handling
4	If major NFCDevVNo is bigger than major T3Vno	The NFC Forum Device might implement the support for previous versions of this specification in addition to its main version. In case the NFC Forum Device supports the previous version, it SHALL access the Type 3 Tag. On the contrary, in case the NFC Forum Device does not support the previous version, it SHALL reject the Type 3 Tag.

Future versions of this specification have to define the allowed actions to an NFC Forum Tag with a version number lower than the version number of the NFC Forum Device (e.g. whether it is allowed to upgrade the NFC Forum Tag to the new version).

6.2 NDEF storage

The NDEF data SHALL be stored on the Type 3 Tag using the memory blocks assigned to the Service with the Service Code 000Bh.

If the Type 3 Tag allows write access to the NDEF data, the Service 0009h is available as an Overlap Service.

Table 8: NDEF Services

Usage	Service setup
NDEF is Read-Only	Service Code 000Bh is available
NDEF is Read/Writeable	Service Code 0009h and Service Code 000Bh are available as Overlap Services

The first byte of Block number 1 of the Service 000Bh contains the first byte of NDEF data. If the NDEF data is larger than 16 bytes, the 17th byte of NDEF data is stored in the first byte of User block number 2. The whole NDEF data is stored on the Type 3 Tag in this way.

When the NFC Forum Device reads NDEF data, it SHALL ignore the padding if the NDEF data does not use all bytes of the last block. When the NFC Forum Device writes NDEF data to the Type 3 Tag, padding data MAY be added if the data length of the NDEF data is not a multiple of 16. Padding byte SHALL be (00h).

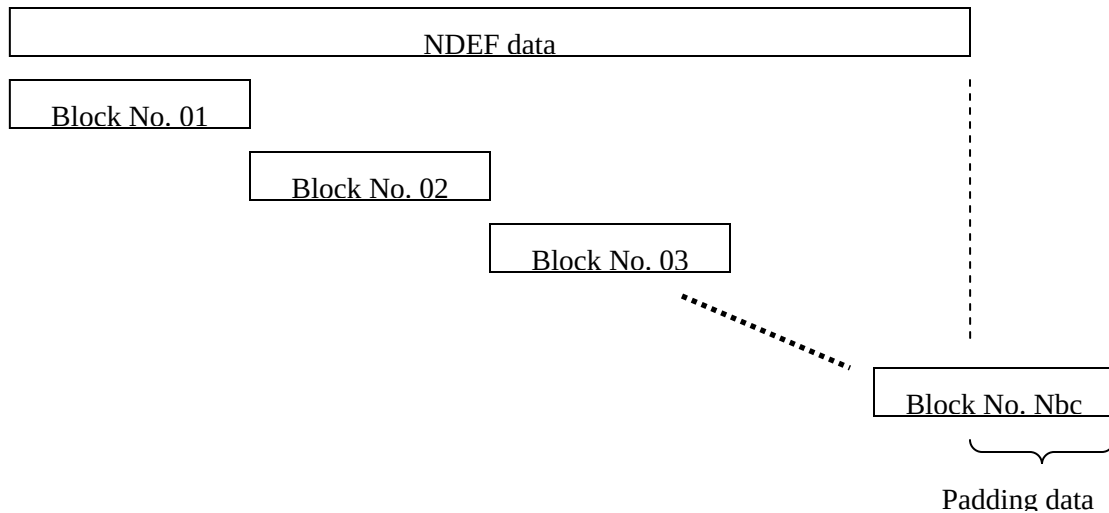


Figure 10: Relation between NDEF data and blocks storing it

6.3 Life cycle

This chapter describes the life cycle of a NDEF enabled Type 3 Tag.

An NDEF enabled Type 3 Tag may be in different states. Every state has its own valid operations.

For all states the Type 3 Tag is considered to be initialized properly for NDEF data (Service Code is 12FCh, the Service 000Bh exists and the attribute information data is initialized properly).

The NDEF enabled Type 3 Tag shall be delivered in one of the following states: INITIALISED, READ/WRITE or READ-ONLY.

The NFC Forum Device shall interpret the Type 3 Tag to be in one of the following states:

- **INITIALIZED**

The Type 3 Tag can be used to write NDEF data. The Service 0009h exists as Overlap Service. The NFC Forum device can write NDEF data using this service code. In this state, the Type 3 Tag does not contain any valid NDEF message content (Ln byte in the Attribute Information Block is 0h).

- **READ/WRITE**

The Type 3 Tag is considered to contain a valid NDEF message and be available for read/write access. The Service 0009h exists as Overlap Service. The NFC Forum device can read and write NDEF data using this service code.

This state provides the ability to read the NDEF message and also to modify, i.e. completely over-write the existing NDEF message with a new NDEF message.

- **READ ONLY**

The Type 3 Tag is considered to contain a valid NDEF message and be available for read only access. The NFC Forum device can read NDEF data using the service code 000Bh. The NDEF message cannot be deleted or overwritten with a new NDEF message. The Service 0009h does not exist.

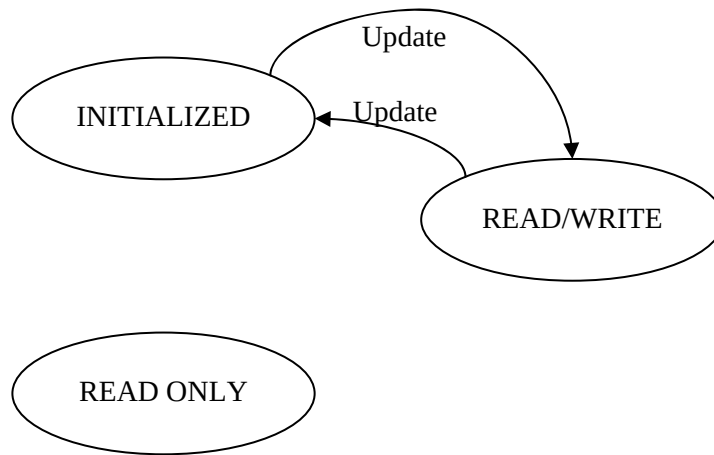


Figure 11: Life cycle states

6.4 Command sequence description

6.4.1 NDEF detection

NDEF data in the Type 3 Tag MAY be detected by an NFC Forum Device as follows:

STEP 1:

The first step to detect NDEF enabled Type 3 Tags consists of the detection of Type 3 Tags in the RF field having a System Code of 12FCh.

The NFC Forum Device sends a Polling command with System Code 12FCh. NDEF enabled Type 3 Tags respond to the Polling command by sending a Polling response including IDm and PMm.

If there is at least one Type 3 Tag with System Code 12FCh in the field, go to STEP 2, otherwise repeat STEP 1.

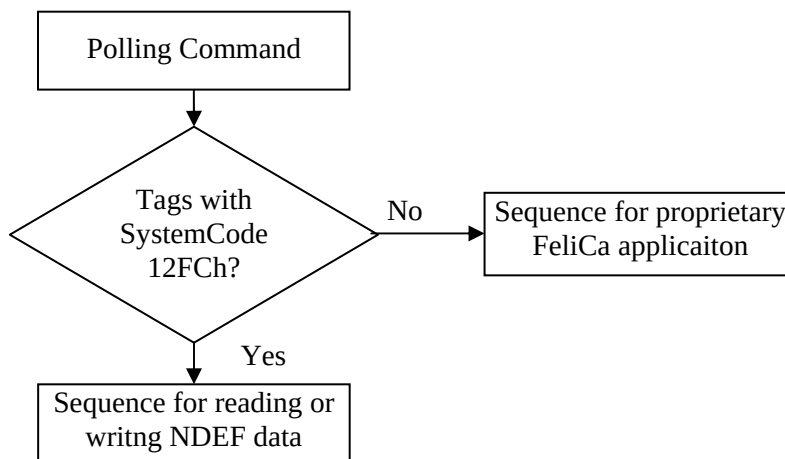


Figure 12: General sequence for detecting NDEF data

STEP 2:

If there are more than one NDEF enabled Type 3 Tags in the field the NFC Forum Device SHALL select one of them for this step. The NFC Forum Device MAY perform step 2 sequentially for each NDEF enabled Type 3 Tag.

NFC Forum Device SHALL read the Attribute Information data using the Check command. For the Check command, the NFC Forum Device SHALL use the same IDm that it received in the Polling response from the Type 3 Tag.

The Attribute Information data contains a checksum to confirm that the Attribution Information data is valid. The NFC Forum Device SHALL validate the checksum. The NFC Forum Device SHALL check if it supports the NDEF mapping version number based on the rules given in chapter 6.1.3.

If the checksum is correct and the NFC Forum Device supports the NDEF mapping version, the NFC Forum Device MAY proceed with

- The read NDEF message phase, or
- The write NDEF message phase

If the checksum is not correct or the NFC Forum Device does not support the NDEF mapping version, the NFC Forum Device SHALL not read or write NDEF data to this Type 3 Tag.

If the Attribute Information data is valid, the NFC Forum Device knows the parameters Nbr, Nrwl, Nmaxb, RWflag and Ln.

If the value of Ln is greater than 0, the Type 3 Tag contains NDEF data.

6.4.2 Read NDEF message

The purpose of this phase is to read the data blocks containing NDEF data in order to assemble the NDEF data structure stored on the Type 3 Tag.

For all commands, the NFC Forum Device SHALL use the same IDm that was used during the NDEF detection phase for the selected Type 3 Tag.

Before reading NDEF data, the NFC Forum Device MAY check if the WriteFlag is OFF which means that previous writing sequences finished correctly. Even if the WriteFlag is ON, the NDEF data on the Type 3 Tag may be valid.

NFC Forum Device SHALL read the blocks from block No. 01 until block number Nbc. Nbc SHALL be calculated as explained in chapter 6.1.2.

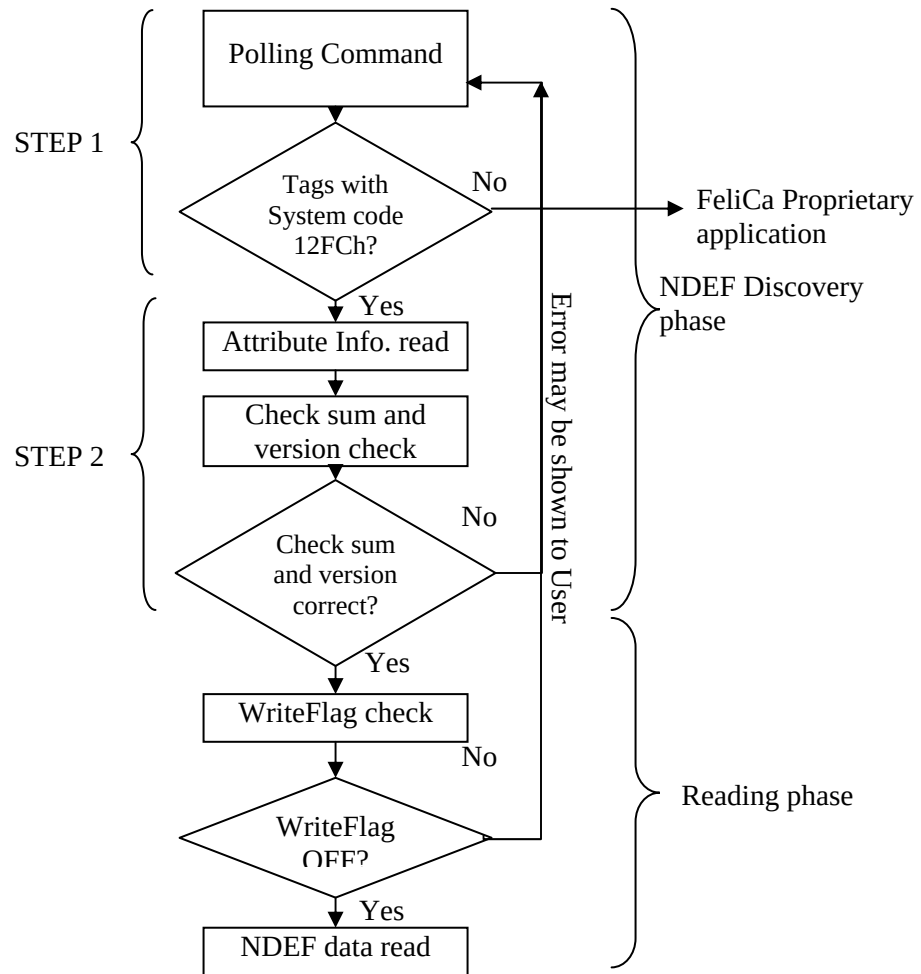


Figure 13: General Sequence for reading NDEF data

6.4.3 Write NDEF message

The purpose of this phase is to write NDEF data to a Type 3 Tag. Any existing NDEF data on the Type 3 Tag will be overwritten.

For all commands, the NFC Forum Device SHALL use the same IDm that was used during the NDEF detection phase for the selected Type 3 Tag.

The NFC Forum Device SHALL check the RWFlag and only attempt to write if its value is 01h.

Before writing NDEF data, the NFC Forum Device MAY check if the NDEF data size is smaller than the maximum memory size for NDEF data available on the Type 3 Tag. In this case the NFC Forum Device SHALL calculate the maximum memory size for NDEF by multiplying Nmaxb times 16.

Before writing NDEF data, the NFC Forum Device SHOULD update the Attribute Information with the WriteFlag set to ON.

The NFC Forum Device SHALL start writing the blocks from Block No. 01 until block number Nbc. The number of blocks to use for NDEF data (Nbc) SHALL satisfy $Nbc \leq Nmaxb$.

After writing the NDEF data, the NFC Forum Device SHALL write updated Attribute Information data to the User Block No. 00 using the Update command. The Length information within the Attribute Information SHALL be changed according to the amount of NDEF data written. The WriteFlag SHALL be set to OFF.

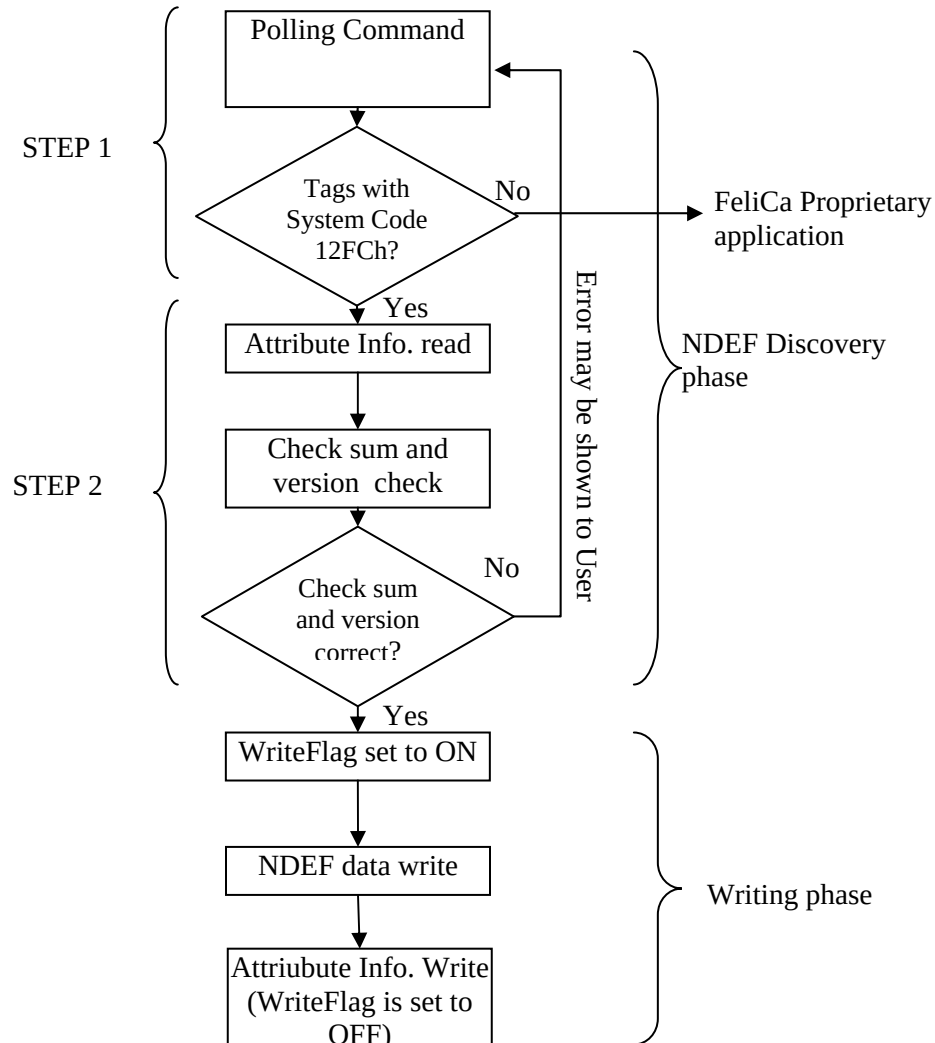


Figure 14: General Sequence for writing NDEF data

6.4.4 State changes

A state change from the 'INITIALIZED' to the 'READ/WRITE' state can be performed by writing a valid NDEF message to the Type 3 Tag as described in chapter 6.4.3.

A state change from the 'READ/WRITE' to the 'INITIALIZED' state can be performed by updating the attribute information block with an Ln value of 0.

A. Typical activation sequence

This is an informative chapter outlining the typical activation sequence from the point of view of a stand-alone NFC Forum Device implementation.

The activation sequence of a Type 3 Tag is compatible with the Initialisation and Single Device Detection used by ISO/IEC 18092 for the 212 kbps and 424 kbps modes (see [ISO/IEC 18092]).

The communication between NFC Forum Device and the Type 3 Tag is initiated as follows:

1. The NFC Forum Device switches on the RF field. The NFC Forum Device sends a Polling command. The NFC Forum Device should wait for at least 5 milliseconds after switching the RF field on before sending a polling command (Remark: According to NFCIP-1, NFC Forum Devices may need up to 2 seconds in the 212/424 kbps modes as activation time).
2. The NFC Forum Device waits to receive Polling responses from the Type 3 Tags in the field. If the NFC Forum Device fails to receive a Polling response, then the NFC Forum Device may send a Polling command again. If there was a collision when receiving the Polling responses, the NFC Forum Device should send a Polling command again.
3. Each Polling response contains the IDm information of the sending Type 3 Tag, which can be used by the NFC Forum Device as part of the Check and Update commands to further communicate with the Type 3 Tag. The NFC Forum Device may communicate with multiple Type 3 Tags using their respective IDm.

B. Revision History

The following table outlines the revision history of Type 3 Tag Operation.

Table 9: Revision History

Document Name	Revision and Release Date	Status	Change Notice	Supersedes
NFCForum-TS-Type-3-Tag_1.0	1.0, July 2007	Final	None	



Type 4 Tag Operation

Technical Specification

NFC Forum™

NFCForum-TS-Type-4-Tag_1.0

2007-03-13

RESTRICTIONS ON USE

This specification is copyright © 2005-2007 by the NFC Forum, and was made available pursuant to a license agreement entered into between the recipient (Licensee) and NFC Forum, Inc. (Licensor) and may be used only by Licensee, and in compliance with the terms of that license agreement (License). If you are not the Licensee, you are not authorized to make any use of this specification. However, you may obtain a copy at the following page of Licensor's Website: http://www.nfc-forum.org/resources/spec_license after entering into and agreeing to such license terms as Licensor is then requiring. On the date that this specification was downloaded by Licensee, those terms were as follows:

1. LICENSE GRANT.

Licensor hereby grants Licensee the right, without charge, to copy (for internal purposes only) and share the Specification with Licensee's members, employees and consultants (as appropriate). This license grant does not include the right to sublicense, modify or create derivative works based upon the Specification.

2. NO WARRANTIES.

THE SPECIFICATION IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY, COMPLETENESS AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL LICENSOR, ITS MEMBERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE SPECIFICATION.

3. THIRD PARTY RIGHTS.

Without limiting the generality of Section 2 above, LICENSOR ASSUMES NO RESPONSIBILITY TO COMPILE, CONFIRM, UPDATE OR MAKE PUBLIC ANY THIRD PARTY ASSERTIONS OF PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS THAT MIGHT NOW OR IN THE FUTURE BE INFRINGED BY AN IMPLEMENTATION OF THE SPECIFICATION IN ITS CURRENT, OR IN ANY FUTURE FORM. IF ANY SUCH RIGHTS ARE DESCRIBED ON THE SPECIFICATION, LICENSOR TAKES NO POSITION AS TO THE VALIDITY OR INVALIDITY OF SUCH ASSERTIONS, OR THAT ALL SUCH ASSERTIONS THAT HAVE OR MAY BE MADE ARE SO LISTED.

4. TERMINATION OF LICENSE.

In the event of a breach of this Agreement by Licensee or any of its employees or members, Licensor shall give Licensee written notice and an opportunity to cure. If the breach is not cured within thirty (30) days after written notice, or if the breach is of a nature that cannot be cured, then Licensor may immediately or thereafter terminate the licenses granted in this Agreement.

5. MISCELLANEOUS.

All notices required under this Agreement shall be in writing, and shall be deemed effective five days from deposit in the mails. Notices and correspondence to the NFC Forum address as it appears below. This Agreement shall be construed and interpreted under the internal laws of the United States and the Commonwealth of Massachusetts, without giving effect to its principles of conflict of law.

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

Contents

1	Introduction.....	1
1.1	Objectives.....	1
1.2	Purpose.....	1
1.3	Applicable Documents or References.....	1
1.4	Administration.....	1
1.5	Special Word Usage.....	2
1.6	Name and Logo Usage.....	2
1.7	Intellectual Property.....	2
1.8	Glossary.....	2
1.9	Convention and notations.....	4
1.9.1	Representation of numbers.....	4
2	Management of the Memory Structure	5
3	RF interface.....	6
4	Framing / Transmission Handling	7
5	Command Set	8
5.1	Activation of the Transmission Protocol.....	8
5.2	High Level Command Set.....	8
5.2.1	Format of C-APDU	8
5.2.2	Format of R-APDU	9
6	NDEF Detection and Access.....	10
6.1	NDEF Management.....	10
6.1.1	Version Treating.....	11
6.1.2	TLV blocks.....	12
6.2	NDEF Storage	15
6.3	Life Cycle.....	15
6.3.1	INITIALISED State	16
6.3.2	READ/WRITE State	16
6.3.3	READ-ONLY State.....	16
6.4	Command Sequence Description	16
6.4.1	NDEF Detection Procedure.....	16
6.4.2	NDEF Tag Application Select Procedure.....	17
6.4.3	Capability Container Select Procedure.....	18
6.4.4	Capability Container Read Procedure.....	19
6.4.5	NDEF Select Procedure.....	20
6.4.6	NDEF Read Procedure	21
6.4.7	NDEF Update Procedure.....	22
6.4.8	State Changes	23

Figures

Figure 1:	Length Field Formats.....	12
Figure 2:	Life Cycle with State Changes.....	23
Figure 3:	NDEF Tag Application Example.....	25

Tables

Table 1: Command Set overview	8
Table 2: Format of C-APDU	9
Table 3: Format of R-APDU	9
Table 4: Data Structure of the Capability Container file.....	10
Table 5: Handling of the mapping document version numbers.....	11
Table 6: Defined TLV blocks.....	12
Table 7: Data Structure of the Proprietary File	14
Table 8: Data Structure of the NDEF file.....	15
Table 9: NDEF Tag Application Select – C-APDU.....	17
Table 10: NDEF Tag Application Select – Detailed C-APDU Field Description	18
Table 11: NDEF Tag Application Select – R-APDU.....	18
Table 12: Capability Container Select – C-APDU.....	18
Table 13: Capability Container Select – Detailed C-APDU Field Description.....	19
Table 14: Capability Container Select – R-APDU.....	19
Table 15: ReadBinary Command – C-APDU	19
Table 16: ReadBinary Command – Detailed C-APDU Field Description.....	20
Table 17: ReadBinary Command – R-APDU	20
Table 18: NDEF Select – C-APDU.....	20
Table 19: NDEF Select– Detailed C-APDU Field Description.....	21
Table 20: NDEF Select – R-APDU.....	21
Table 21: NDEF Update – C-APDU	22
Table 22: NDEF Update– Detailed C-APDU Field Description.....	22
Table 23: NDEF Update data structure – R-APDU	22
Table 24: CC File Example	26
Table 25: NDEF File Example.....	26
Table 26: Revision History.....	31

1 Introduction

This specification is part of the NFC Forum documentation about tag types that an NFC Forum device needs to support in reader/writer mode.

This specification documents how an NFC Forum Device SHALL operate an NFC Forum Type 4 tag platform. This is not a specification of the NFC Forum Type 4 tag platform itself.

1.1 Objectives

The purpose of this specification is to document the requirements and to specify, with a set of rules and guidelines, the NFC Forum Device operation and management of the Type 4 tag platform.

This specification assumes that the Collision Detection and Device Activation activities have been performed as documented in the Mode Switch specifications [DIGPROT] and [ANINT].

This specification also defines the data mapping and how the NFC Forum Device detects, reads, and writes NDEF data into the Type 4 tag platform in order to achieve and maintain interchangeability and interoperability.

1.2 Purpose

The purpose of this specification is to document the requirements and to specify, with a set of rules and guidelines, the NFC Forum Device operation and management of a Type 4 tag platform.

This specification also defines the data mapping and how the NFC Forum Device detects, reads, and writes NDEF data into the Type 4 tag platform in order to achieve and maintain interchangeability and interoperability.

1.3 Applicable Documents or References

[ANINT]	NFC Analogue Interface Specification.
[DIGPROT]	NFC Digital Protocol Specification.
[ISO 7816-4]	ISO/IEC 7816-4 Identification cards - Integrated circuit cards - Organization, security and commands for interchange.
[NDEF]	“NFC Data Exchange Format (NDEF)” NFC Forum™, May 2006.
[RFC 2119]	S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Harvard University, March 1997.

1.4 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.

1.5 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 Type 4 Tag Operation 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

APDU

Application Protocol Data Unit

C-APDU

Command APDU

CC

Capability Container

DF

Directory File

<i>EF</i>	Elementary File (file identifier)
<i>Lc</i>	Length command
<i>Le</i>	Length expected
<i>LSB</i>	least significant byte
<i>lsb</i>	least significant bit
<i>MLc</i>	Maximum data Length C-APDU
<i>MLe</i>	Maximum data Length R-APDU
<i>MSB</i>	most significant byte
<i>msb</i>	most significant bit
<i>NDEF</i>	NFC Data Exchange Format
<i>NFC Forum device</i>	A device that supports the following modus operandi: Initiator, Target and Reader/Writer. It may also support Card Emulator. NFC Forum compliant Device. In this document the NFC Forum device is always using the Reader/Writer modus operandi (for more information see [DIGPROT]).
<i>R-APDU</i>	Response APDU
<i>RF</i>	Radio Frequency
<i>RFU</i>	Reserved for future use
<i>Type 4 tag platform</i>	A legacy platform supporting a subset of a Technology (also called Technology Subset). Type 4 tag platform, which uses a particular subset of NFC – Type A technology or NFC- Type B technology, including anticollision (for more information see [DIGPROT]).

1.9 Convention and notations

1.9.1 Representation of numbers

The following conventions and notations apply in this document unless otherwise stated.

Binary numbers are represented by strings of digits 0 and 1 shown with the most significant bit (msb) left and the least significant bit (lsb) right, “b” is added at the end.

Example: 11110101b

Hexadecimal numbers are represented is using the the numbers 0 - 9 and the characters A – F, a “h” is added at the end. The Most Significant Byte (MSB) is shown on the left, the Least Significant Byte (LSB) on the right.

Example: F5h

Decimal numbers are represented as is (without any tailing character).

Example: 245

2 Management of the Memory Structure

The Type 4 tag platform contains at least the NDEF Tag Application. The NDEF Tag Application contains the NDEF messages on a Type 4 tag platform providing a file system composed of at least two EF files (see [ISO 7816-4]): the Capability Container file (CC file, see section 6.1) and the NDEF file (NDEF file, see section 6.2).

Concerning the EF files, the byte with offset value equal to zero is the Most Significant Byte (MSB), and the byte with the highest offset value is the Least Significant Byte (LSB).

In the definition of this document if not otherwise specified, the bit and byte ordering when defining packets and messages follows the big-endian byte order.

3 RF interface

The RF interface of the NFC Forum device is defined in [ANINT]. The NFC Forum device SHALL comply with the RF interface as defined in the relevant clauses of [ANINT].

4 Framing / Transmission Handling

This chapter describes the framing (also called packet structures), and the transmission handling of the NFC Forum device.

The NFC Forum device SHALL comply with the bit coding, the character coding, the frame coding, the byte coding and as well the commands and responses up to and including the activation sequence of the Type 4 tag as defined in [DIGPROT].

The NFC Forum device SHALL comply with the half duplex protocol (see Type 4 tag in [DIGPROT]) for exchanging the commands and responses defined in chapter 5.

5 Command Set

This chapter describes the command set of the NFC Forum Device.

5.1 Activation of the Transmission Protocol

The activation of the transmission protocol is described in Type 4 tag section of [DIGPROT].

5.2 High Level Command Set

The commands that SHALL be supported at ISO/IEC 7816-4 [ISO 7816-4] by the NFC Forum device are listed in Table 1. The format of the commands and the relative responses of Table 1 are described in section 5.2.1 and 5.2.2. To detect and access the NFC Forum data, the specific settings of the command and response fields are described in chapter 6.

The commands of the NFC Forum device SHALL support short length fields and are compatible with the commands defined in [ISO 7816-4].

The standard [ISO 7816-4] names: command APDU (C-APDU) the commands sent from the NFC Forum device, and responses APDU (R-APDU) the responses to a specific command received by the NFC Forum Device.

NOTE [ISO 7816-4] defines a whole range of commands where only a few are relevant for a Type 4 tag platform implementation.

Table 1: Command Set overview

Command/Response	Description
Select	Selection of applications, or files
ReadBinary	Read data from file
UpdateBinary	Update (erase and write) data to file

NOTE This specification provides means of reading and writing the NDEF file. It does not cover the personalization of the Type 4 tag platform and modifications of access rights. It is assumed that the Type 4 tag platform has already been personalized as expected.

5.2.1 Format of C-APDU

The C-APDU is sent by an NFC Forum device as it is defined in accordance to [ISO 7816-4]. The following part is intended to illustrate the activation of the transmission protocol.

The length fields Lc and Le as well as the data field are optional. Short length field (one byte long) as defined in [ISO 7816-4] SHALL be supported.

Table 2 describes the format of the C-APDU. In chapter 6 the structure of Table 2 is used to describe the different C-APDU.

Table 2: Format of C-APDU

CLA	INS	P1	P2	Lc	Data	Le
Class byte	Instr. Byte	Param. byte 1	Param. byte 2	Lc field	Data bytes (Lc bytes)	Le field

Class Byte: the Class byte SHALL be set 00h (compliant to [ISO 7816-4], no secure messaging).

Instruction Byte: the Instruction byte indicates the command to process.

Parameter Byte 1: the parameter SHALL be set to 00h if no value is specified for instruction use.

Parameter Byte 2: the parameter SHALL be set to 00h if no value is specified for instruction use.

Data Field Length Lc: the data field length Lc is optional. If it is present, it SHALL contain the number of bytes in the data field of the command, and it SHALL be different from zero.

Data Field: the data field is optional.

Expected Response Length Le: the expected response length Le is optional. If it is present, the response R-APDU (see section 5.2.2) contains the number of expected bytes.

5.2.2 Format of R-APDU

The R-APDU is the response to the C-APDU received by NFC Forum device. It is defined in accordance to [ISO 7816-4]. The following part is intended to illustrate the activation of the transmission protocol.

Table 3 describes the format of the C-APDU. In chapter 6 the structure of Table 3 is used to describe the different C-APDU.

Table 3: Format of R-APDU

Response Body	SW1	SW2
Data bytes	Status Word 1	Status Word 2

Response Body: the response body is optional and carries the data of the R-APDU.

Response Status Bytes: the response status bytes SW1 and SW2 are mandatory. The coding of the response status bytes follow the coding as defined in [ISO 7816-4].

6 NDEF Detection and Access

This section describes how the NFC Forum defined data are stored and accessed by the NFC Forum device.

6.1 NDEF Management

To detect and access NFC Forum defined data the NFC Forum device retrieves and uses the Capability Container (CC) file contained inside the NDEF Tag Application. The CC file contains management data, and it is stored inside a read-only EF file, see [ISO 7816-4]. The NFC Forum device SHALL accept NDEF Tag Application having the CC file with file identifier equal to E103h.

The data structure of the CC file is described in Table 4. The CC file SHALL contain from offset 0000h to 0006h the fields: CCLen, Mapping Version, MLe and MLc. One NDEF File Control TLV SHALL be present at the offset 0007h. Zero, one or more TLV blocks MAY be present from offset 000Fh.

If not specified otherwise, the term NDEF file in the following chapters refers to the NDEF file indicated by the NDEF File Control TLV stored at the offset 0007h in the CC file.

Table 4: Data Structure of the Capability Container file

Offset (bytes)	Size (bytes)	Field	Remarks
0000h	2	CCLen (bytes)	Indicates the size of this capability container (including this field). Valid CCLen values are between 000Fh and FFFEh, FFFFh and 0000h-000Eh are reserved for future use.
0002h	1	Mapping Version	Indicates the mapping specification version it is compliant to, see section 6.1.1. The most significant nibble (the 4 most significant bits) SHALL indicate the major version number, and the least significant nibble (the 4 least significant bits) SHALL indicate the minor version number.
0003h	2	MLe (bytes); Maximum R-APDU data size	This value defines the maximum data size that can be read from the Type 4 tag using a single ReadBinary command. The valid values are MLe = 000Fh-FFFFh. The values between 0000h-0000Eh are reserved for future use.
0005h	2	MLc (bytes); Maximum C-APDU data size	This value defines the maximum data size that can be sent to the Type 4 tag using a single UpdateBinary command. The valid range is MLc = 0001h-FFFFh bytes. The value 0000h is RFU.
0007h	8	NDEF File Control TLV	TLV block that contains information to control and manage the NDEF file, see section 0
000Fh	-	TLV Blocks	Zero, one or more TLV blocks MAY be starting from offset Fh

6.1.1 Version Treating

The Mapping Version field in the CC contains the version of the applied mapping document to the NFC Forum Type 4 tag platform. The mapping document version SHALL be indicated with two numbers: major number version (most significant nibble) and minor version number (least significant nibble).

This document specifies the following mapping version: major version number equal to 1h and minor version number equal to 0h i.e. mapping version 1.0.

The handling rules of the different mapping document version numbers applied to the Type 4 tag platform (called T4VNo) and the one implemented in the NFC Forum device (called NFCDevVNo) is explained in the 4 cases of Table 5.

Table 5: Handling of the mapping document version numbers

No	Version Number Case	Handling Rules
1	Major NFCDevVNo is equal to major T4VNo, and minor NFCDevVNo is bigger than or equal to minor T4VNo	The NFC Forum device SHALL access the Type 4 tag and SHALL use all features of the applied mapping document to this Type 4 tag platform.
2	If major NFCDevVNo is equal to major T4VNo, and minor NFCDevVNo is lower than minor T4VNo	Possibly not all features of the Type 4 tag platform can be accessed. The NFC Forum device SHALL use all its features and SHALL access this Type 4 tag platform.
3	If major NFCDevVNo is smaller than major T4VNo	Incompatible data format. The NFC Forum device cannot understand the Type 4 tag platform data. The NFC Forum device SHALL reject this Type 4 tag platform.
4	If major NFCDevVNo is bigger than major T4VNo	The NFC Forum device MAY implement the support for previous versions of this specification in addition to its main version. In case the NFC Forum device has the support from previous version, it SHALL access the Type 4 tag platform. On the contrary, in case the NFC Forum device has not the support from previous version, it SHALL reject the Type 4 tag platform.

NOTE Future versions of this specification have to define the allowed actions to an NFC Forum Type 4 tag platform with a version number lower than the version number of the NFC Forum device (e.g. whether it is allowed to upgrade the tag to the new version).

6.1.2 TLV blocks

A TLV block consists of one to three fields:

- *T* The tag field identifies the type of the TLV block, see Table 6, and consists of a single byte encoding a number from 00h to FEh. The tag field values from 00h to 03h and from 06h to FFh are reserved for future use by the NFC Forum.
- *L* The length field provides the size in bytes of the value field. It has two different formats composed of one or three bytes. The NFC Forum device SHALL understand all two length field formats. Figure 1 shows the two different length field formats. However depending on the tag field value, the length field may not be present.
 - *One byte format:* The NFC Forum device SHALL use the one byte format to code the length of the value field between 00h and FEh bytes. The NFC Forum device SHALL interpret this byte as a cardinal if the value is between 00h and FEh. If it contains FFh, the NFC Forum device SHALL interpret the value as flag that specifies that the length field is composed of more than one byte.
 - *Three consecutive bytes format:* The NFC Forum device SHALL use this format to code the length of the value field between 00FFh and FFFEh bytes. The first byte is assumed to be a flag equal to FFh indicating that two more bytes are present. The NFC Forum device SHALL interpret the two more bytes as a word. The NFC Forum device SHALL interpret this word as a cardinal if the value is between 00FFh and FFFEh. The value FFFFh is reserved for future use (RFU).

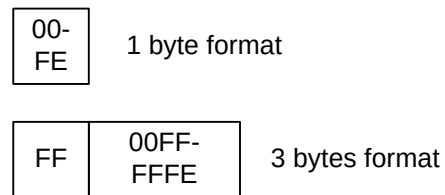


Figure 1: Length Field Formats

- *V* If the length field is equal to 00h or there is no length field, the value field is not present, i.e. the TLV block is empty. If there is the length field and indicates a length of the value field *N* bigger than zero ($N > 0$), the value field consists of *N* consecutive bytes.

Table 6 lists the TLV blocks specified by this document that are described in the following sections.

Table 6: Defined TLV blocks

TLV block name	Tag Field Value	Short Description
NDEF File Control TLV	04h	It contains control information concerning the EF file containing the NDEF message
Proprietary File Control TLV	05h	Contains control information concerning a 'Proprietary file', which is a EF file containing proprietary data

NFC Forum Devices SHALL ignore and jump over those TLV blocks that make use of reserved tag field values. To jump over a TLV block with reserved tag field values, the NFC Forum device SHALL read the length field to understand the length of the value field.

NOTE Future definitions of TLV blocks composed of only the tag field are not backward compatible with this NFC Forum specification.

6.1.2.1 NDEF File Control TLV

The NDEF File Control TLV is always present inside the CC file. It provides control information about the EF file containing the NDEF message, see section 6.2. The NFC Forum device SHALL be able to read and process the NDEF File Control TLV. The NFC Forum device SHALL check that the CC file contains an NDEF File Control TLV at the offset 0007h.

Below the encoding of the 3 TLV fields of NDEF File Control TLV are shown:

- T is equal to 04h (see Table 6).
- L is equal to 06h.
- V is composed of 6 bytes that specifies size, read access conditions, write access conditions, and EF identifier of the EF file containing the NDEF message. The 6 bytes are encoded in the following way:
 - File Identifier, 2 bytes. It indicates a valid NDEF file. The valid range is 0000h-FFFEh, except the values E102h, E103h, 3F00h, and 3FFFh. The values 3F00h, 3FFFh are reserved and FFFFh is reserved for future use (see [ISO 7816-4]).
 - Maximum NDEF file size, 2 bytes. Maximum size in bytes of the NDEF file. This size does not reflect the size of the contained NDEF message as such but rather the size of the created file containing the NDEF message. The valid range is 0005h-FFFEh. The values 0000h-0004h, and FFFFh are reserved for future use.
 - NDEF file read access condition, 1 byte:
 - The value 00h indicates read access granted without any security,
 - The values from 01h to 7Fh and FFh are reserved for future use, and
 - The values from 80h to FEh are proprietary.
 - NDEF file write access condition, 1 byte:
 - The value 00h indicates write access granted without any security.
 - The value FFh indicates no write access granted at all (read-only).
 - The values from 01h to 7Fh are reserved for future use, and
 - The values from 80h to FEh are proprietary.

NOTE The maximum size of the NDEF file is not limited by the Maximum NDEF file size but Offset and Length Le fields of Read Binary and NDEF Update C-APDUs (see Table 15, Table 16, Table 21 and Table 22). The maximum size of the NDEF file is so reduced to $7FFFh + FFh = 80FEh$ bytes.

6.1.2.2 Proprietary File Control TLV

The Proprietary File Control TLV contains control information about the Proprietary file. The Proprietary file is an EF file that contains proprietary information (see below). The CC file contains zero, one or more Proprietary File Control TLV. The NFC Forum device SHALL be able to read/process this TLV block. The NFC Forum device MAY ignore the data contained in the Proprietary File Control TLV. Below the encoding of the 3 TLV fields of Proprietary TLV are shown:

- T is equal to 05h (see Table 6).
- L is equal to 06.
- V is composed of 6 bytes that specifies size, read access conditions and write access conditions, and EF identifier of the EF file containing the proprietary data. The 6 bytes are encoded in the following way:
 - File Identifier, 2 bytes. It indicates a valid Proprietary file. The valid range is 0000h-FFFEh, except the values E102h, E103h, 3F00h, and 3FFFh. The values 3F00h, 3FFFh are reserved and FFFFh are reserved for future use (see [ISO 7816-4]).
 - Maximum Proprietary file size, 2 bytes. Maximum size in bytes of the Proprietary file. The valid range is 0003h-FFFEh. The value FFFFh is reserved for future use.
 - Proprietary file read access condition, 1 byte:
 - The value 00h indicates read access granted without any security,
 - The values from 01h to 7Fh and FFh are reserved for future use, and
 - The values from 80h to FEh are proprietary.
 - Proprietary file write access condition, 1 byte:
 - The value 00h indicates write access granted without any security.
 - The value FFh indicates no write access granted at all (read-only).
 - The values from 01h to 7Fh are reserved for future use, and
 - The values from 80h to FEh are proprietary.

The proprietary data is stored inside an EF file (see [ISO 7816-4]) called Proprietary file using the data structure described in Table 7. An NDEF Tag Application can have zero, one or more Proprietary files.

Table 7: Data Structure of the Proprietary File

Offset (bytes)	Size (bytes)	Field	Remarks
0000h	2	PLEN [bytes]	The Proprietary Length field (PLEN) indicates the size of the proprietary data stored in the Proprietary file. Valid PLEN values are between 0000h and FFFEh, FFFFh is reserved for future use.
0002h	x	proprietary data	proprietary data

NOTE x indicates the size of the proprietary data.

6.2 NDEF Storage

The data format of the NDEF message is defined in [NDEF]. The NDEF message is stored inside an EF file (see [ISO 7816-4]) called NDEF file using the data structure described in Table 8. The NFC Forum device SHALL check that the NDEF file specified in the mandatory NDEF File Control TLV is present in the NFC Forum application (see section 6.4.1).

Table 8: Data Structure of the NDEF file

Offset	Size	Field	Remarks
0h	2	NLEN [bytes]	The NDEF Length field (NLEN) indicates the size of the NDEF message stored in the NDEF file. Valid NLEN values are between 0000h and FFFEh, FFFFh is reserved for future use.
2h	x	NDEF message	NDEF message, see [NDEF].

NOTE x indicates the size of the NDEF message.

The NDEF file contains either an empty or a non-empty NDEF message. Below the definition of an empty NDEF message is given in Appendix A.

6.3 Life Cycle

The NFC Forum device classifies a Type 4 tag platform into several states. The state is reflected by the content of the NDEF Tag Application. Every state has its own valid operations.

The transition states are only relevant for NFC Forum devices, which are capable of writing Type 4 tag platforms.

The NFC Forum device SHALL detect and accept a Type 4 tag platform in one of the following states: INITIALISED, READ/WRITE and READ-ONLY.

The NFC Forum device SHALL ignore Type 4 tag platform not in a valid state. The reasons MAY be:

- The NDEF Tag Application, the CC file, or the NDEF file is not present in the Type 4 tag platform.
- Misconfigured CC file.
- Not allowed write operation on the NDEF file, if Type 4 tag platform is in READ/WRITE state and no other error is detected.
- Misconfigured NDEF file.

6.3.1 INITIALISED State

In this state the NFC Forum device MAY modify the content of the NFC Forum defined data i.e. NDEF Message TLV in the Type 4 tag platform.

The NFC Forum device SHALL detect a Type 4 tag platform in INITIALISED state when:

- The CC file is set as described in section 6.1,
- The NDEF file, indicated by the File Identifier of the NDEF File Control TLV of the CC file at the offset 0007h, is open for read and write access, and
- The NLEN field of the NDEF file is equal to 0000h.

Having detected the INITIALISED state, the NFC Forum device MAY modify the content of the NDEF file.

6.3.2 READ/WRITE State

In this state the NFC Forum device MAY modify the content of the NFC Forum defined data i.e. NDEF Message TLV in the Type 4 tag platform.

The NFC Forum device SHALL detect a Type 4 tag platform in READ/WRITE state when:

- The CC file is set as described in section 6.1,
- The NDEF file, indicated by the File Identifier of the NDEF File Control TLV of the CC file at the offset 0007h, is open for read and write access, and
- The NDEF file contains the NLEN field different from 0000h.

Having detected the READ/WRITE state, the NFC Forum device MAY modify the content of the NDEF file.

6.3.3 READ-ONLY State

In this state the NDEF file is set to read-only.

The NFC Forum device SHALL detect a Type 4 tag in READ-ONLY state when:

- The CC file is set as described in section 6.1,
- The NDEF file, indicated by the File Identifier of the NDEF File Control TLV of the CC file at the offset 0007h, is READ-ONLY state, and
- The NDEF file contains the NLEN field different from 0000h.

6.4 Command Sequence Description

In this section several procedures are described to manage NFC Forum defined data e.g. NDEF message. Finally the different state changes or transitions between the states of the life cycle (see section 6.3) are shown in detail.

6.4.1 NDEF Detection Procedure

The NFC Forum device SHALL use the NDEF detection procedure to detect the NDEF message inside a Type 4 tag platform.

The NDEF file that is found by the NDEF detection procedure is also called mandatory NDEF file. The mandatory NDEF file is always indicated by the NDEF File Control TLV located at the offset 0007h of the CC file.

The NDEF detection procedure is the following:

1. Select the NDEF Tag Application (see section 6.4.2).
2. If the NDEF Tag Application is successfully selected then go to item 3. Otherwise the Type 4 tag platform is not in a valid state.
3. Select the Capability Container (CC) file (see section 6.4.3).
4. If the CC file is successfully selected then go to item 5. Otherwise the Type 4 tag platform is not in a valid state.
5. Read the CC file (see section 6.4.4), and select the NDEF file (see section 6.4.5).
6. If the CC file is successfully read, and the NDEF file has read access without any security, and is successfully selected then go to item 7. Otherwise the Type 4 tag platform is not in a valid state.
7. Read NLEN (NDEF length) from NDEF file (see section 6.4.6):
 1. If $NLEN > 0000h$ and $NLEN \leq \text{Maximum NDEF file size} - 2$, the NDEF message is detected inside the Type 4 tag platform.,
 2. If NLEN is equal to 0000h, no NDEF message is detected in the Type 4 tag platform. The Type 4 tag platform might be in INITIALISED state, and
 3. If NLEN is bigger than Maximum NDEF size-2, the Type 4 tag platform is not in a valid state.

NOTE The NDEF detection procedure does not relate to a valid NDEF message (see [NDEF]). It reads the length of the store data from the NLEN field and does not parse the data itself from the NDEF message field.

6.4.2 NDEF Tag Application Select Procedure

The NFC Forum device SHALL execute the NDEF Tag Application select procedure to select the NDEF Tag Application.

To perform the NDEF Tag Application select procedure, the NFC Forum device SHALL send the Select command (see Table 1) in addition to the sequence defined in section 5.1.

The command parameter of the Select command SHALL be set to select by name. Only when this command returns command completed in the R-APDU the NDEF Tag Application is selected. File control information that is possibly returned MAY be ignored.

The following table defines the C-APDU of the Select command to select the NDEF Tag Application (called NDEF Tag Application Select).

Table 9: NDEF Tag Application Select – C-APDU

CLA	INS	P1	P2	Lc	Data	Le
00h	A4h	04h	00h	07h	D2760000850100h	-

The table below provides a detailed description of the C-APDU fields.

Table 10: NDEF Tag Application Select – Detailed C-APDU Field Description

Byte	Data	Remarks
P1	04h	Select by name
P2	00h	First or only occurrence
Le	-	This field SHALL NOT be present

Table 11: NDEF Tag Application Select – R-APDU

Data	SW1	SW2	Remarks
File control information MAY be returned	90h	00h	Command completed, it is optional to return file control information
-	6Ah	82h	NDEF Tag Application not found, no data is returned

NOTE For further return codes and definitions of those refer to [ISO 7816-4].

6.4.3 Capability Container Select Procedure

The NFC Forum device SHALL perform the Capability Container select procedure to select the capability container (CC) file using the Select command (see Table 1). The command parameter of the Select command SHALL be set to select by elementary file (EF).

Only when this command returns command completed in the R-APDU the CC file is selected. File control information that is possibly returned in the R-APDU MAY be ignored.

The following table defines the C-APDU of the Select command to select the CC file (called Capability Container Select).

Table 12: Capability Container Select – C-APDU

CLA	INS	P1	P2	Lc	Data	Le
00h	A4h	00h	00h	02h	E103h	-

The following table provides a detailed description of the C-APDU fields.

Table 13: Capability Container Select – Detailed C-APDU Field Description

Byte	Data	Remarks
P1	00h	Select by file identifier
P2	00h	First or only occurrence
Lc	02h	2 bytes in data field
Data	E103h	File identifier (EF) of the capability container
Le	-	This field SHALL NOT be present

Table 14: Capability Container Select – R-APDU

Data	SW1	SW2	Remarks
File control information MAY be returned	90h	00h	Command completed, it is optional to return file control information
-	6Ah	82h	Capability container not found, no data is returned

NOTE For further return codes and definitions of those refer to [ISO 7816-4].

6.4.4 Capability Container Read Procedure

The NFC Forum device SHALL use the Capability Container read procedure to read the data from the Capability Container (CC) file after having previously selected it (see section 6.4.3).

The Capability Container read procedure is the following:

1. Read 15 bytes of the CC file (see Table 4) with offset zero in the ReadBinary command (see Table 1).
2. If CLEN<000Fh or read access without any security to the CC file is not granted, the CC file is not valid and the Type 4 tag platform is not in a valid state of the life cycle.

The following table defines the ReadBinary command.

Table 15: ReadBinary Command – C-APDU

CLA	INS	P1	P2	Lc	Data	Le
00h	B0h	Offset		-	-	Length Le

The following table provides a detailed description of the C-APDU fields.

Table 16: ReadBinary Command – Detailed C-APDU Field Description

Byte	Data	Remarks
P1/P2	Offset	File offset where to start reading data, valid range is 0000h-7FFFh
Le	Length Le	The number of bytes to be read from file. The valid range is 01h to FFh.
Lc	-	This field SHALL NOT be present
Data	-	This field SHALL NOT be present.

Table 17: ReadBinary Command – R-APDU

Data	SW1	SW2	Remarks
Content read	90h	00h	Command Completed

NOTE For further return codes and definitions of those refer to [ISO 7816-4].

The NFC Forum device MAY read the data of the CC file after the offset 000Fh using one or more ReadBinary command.

6.4.5 NDEF Select Procedure

The NFC Forum device SHALL use the NDEF select procedure to select the NDEF file using the Select command (see Table 1). The parameter File ID of the Select command SHALL be equal to the File Identifier of the NDEF File Control TLV contained in the CC file at the offset 0007h.

The NFC Forum device successfully selects an NDEF file when the status in the R-APDU is equal to command completed. The NFC Forum device MAY ignore control information that is possibly returned.

NOTE In case the Type 4 tag platform supports an ISO file system the NDEF file is located in the same DF of the CC file.

The NDEF select procedure MAY be done directly after selecting and reading the CC file (see section 6.4.3 and 6.4.4).

The following table defines the Select command to select the NDEF file (called NDEF Select).

Table 18: NDEF Select – C-APDU

CLA	INS	P1	P2	Lc	Data	Le
00h	A4h	00h	00h	02h	File ID	-

The following tables provide a detailed description of the C-APDU fields.

Table 19: NDEF Select– Detailed C-APDU Field Description

Byte	Data	Remarks
P1	00h	Select by file identifier
P2	02h	First or only occurrence
Lc	02h	2 bytes in data field
Data	File ID	File identifier of the NDEF file indicated in the homonymous field of the CC file (see Table 4)
Le	-	This field SHALL NOT be present

Table 20: NDEF Select – R-APDU

Data	SW1	SW2	Remarks
File control information MAY be returned	90h	00h	Command completed, it is optional to return file control information
-	6Ah	82h	NDEF file not found, no data is returned

NOTE For further return codes and definitions of those refer to [ISO 7816-4].

6.4.6 NDEF Read Procedure

The NFC Forum device SHALL execute the NDEF read procedure to read the NDEF file.

The NFC Forum device SHALL do the following operations before reading the NDEF file:

1. Detect successfully the NDEF file using the NDEF detection procedure (see section 6.4.1),
2. Check that the read access without any security is granted for the NDEF file from the information provided by NDEF File Control TLV in the CC file at the offset 0007h (see Table 4 and section 6.1.2.1), and
3. Select the NDEF file (see section 6.4.5).

The NDEF read procedure is the following:

1. Read the NLEN (NDEF length) field of NDEF file (see Table 8) using the ReadBinary command starting from offset zero. The NLEN value MAY be also retrieved from the NDEF detection procedure (see section 6.4.1).
2. Read the NDEF message that starts at offset 0002h of the NDEF file using one or more ReadBinary commands.

The details of the ReadBinary command are described in section 6.4.4.

NOTE The read access without any security is only granted when the NDEF file read access condition indicated in the CC file is set to 00h.

6.4.7 NDEF Update Procedure

The NFC Forum device SHALL execute the NDEF update procedure to write or update an NDEF message inside the NDEF file.

The NFC Forum device SHALL do the following operations before the NDEF update procedure:

1. Detect successfully the NDEF message using the procedure in section 6.4.1,
2. Check that the write access without any security is granted for the NDEF file from the information provided by NDEF File Control TLV in the CC file at the offset 0007h (see Table 4 and section 6.1.2.1), and
3. Select the NDEF file (see section 6.4.5).

The following table defines the NDEF update command (see Table 1) to write or to update the NDEF message inside the NDEF file.

Table 21: NDEF Update – C-APDU

CLA	INS	P1	P2	LC	Data	Le
00h	D6h	Offset		Length Lc	Data to be written in the NDEF file	-

The following table provides a detailed description of the C-APDU fields.

Table 22: NDEF Update– Detailed C-APDU Field Description

Byte	Data	Remarks
P1/P2	Offset	Offset in NDEF file where starting to write data. The valid range is 0000h-7FFFh, see [ISO4]
LC	Length Lc	The number of bytes written to NDEF file. The valid range is 01h to FFh.
Le	-	This field SHALL NOT be present.

Table 23: NDEF Update data structure – R-APDU

Data	SW1	SW2	Remarks
-	90h	00h	Command Completed, no data is returned

NOTE For further return codes and definitions of those refer to [ISO 7816-4].

The NDEF update procedure is the following:

1. If the length of the NDEF message (to be written) is bigger than Maximum NDEF size-2 (see NFC File Control TLV in the section 6.1.2.1) the NDEF update procedure is aborted. Otherwise go to item 2.
2. Write the value 0000h in the NLEN field (see Table 8) using the NDEF Update command,
3. Write the NDEF message in the NDEF message field (see Table 8) using one or more NDEF Update commands, and
4. Write the length of the NDEF message in the NLEN field (see Table 8) using the NDEF Update command.

The NFC Forum device MAY apply this procedure directly after the NDEF read procedure (see section 6.4.6) has been executed.

The item 2 to 3 MAY be done using a single NDEF Update command if the NLEN field and the NDEF message field fit inside the data field of the NDEF Update command.

6.4.8 State Changes

This section describes the possible state changes or also called transition performed by the NFC Forum device. Figure 2 shows the states and the transitions.

The transition covered by this specification is from INITIALISED to READ/WRITE.

NOTE A Type 4 tag platform can be issued in any valid state. A Type 4 tag platform can be issued in READ/WRITE state or even in READ-ONLY state having a predefined NDEF message stored on it.

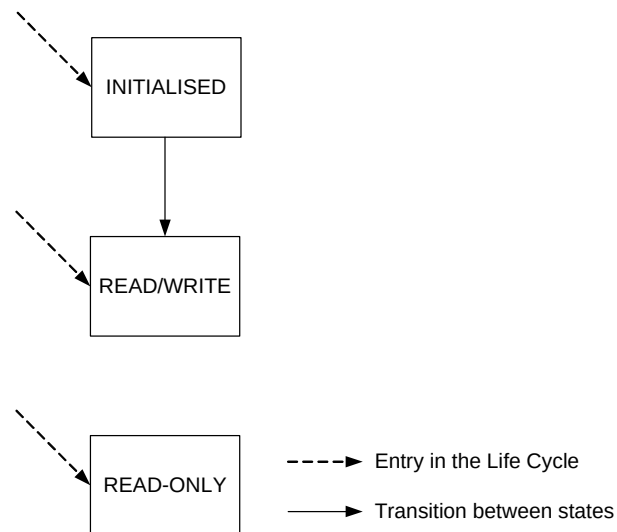


Figure 2: Life Cycle with State Changes

The NFC Forum device SHALL do the following operation to perform the transition from INITIALISED to READ/WRITE: an NDEF message (see [NDEF]) is written in the NDEF Message field of the mandatory NDEF file using the NDEF update procedure (see section 6.4.7).

The NFC Forum device MAY replace a non-empty NDEF message with an empty NDEF message (see Appendix A).

A. Empty NDEF Message

An empty NDEF message (see [NDEF]) is defined as an NDEF message composed of one NDEF record. The NDEF record uses the NDEF short-record layout (SR=1b) with: Type Name Format (TNF) field value equal to 00h (empty, TYPE_LENGTH=00h, PAYLOAD_LENGTH=00h), no ID_LENGTH field (IL=0b), MB=1b, ME=1b, CF=0b. The empty NDEF record (i.e. the empty NDEF message) is composed of 3 bytes and it is equal to D00000h.

B. Example of NDEF Tag Application

This appendix describes from the NFC Forum device point of view an example of NDEF Tag Application (see Figure 3) stored inside a Type 4 tag platform.

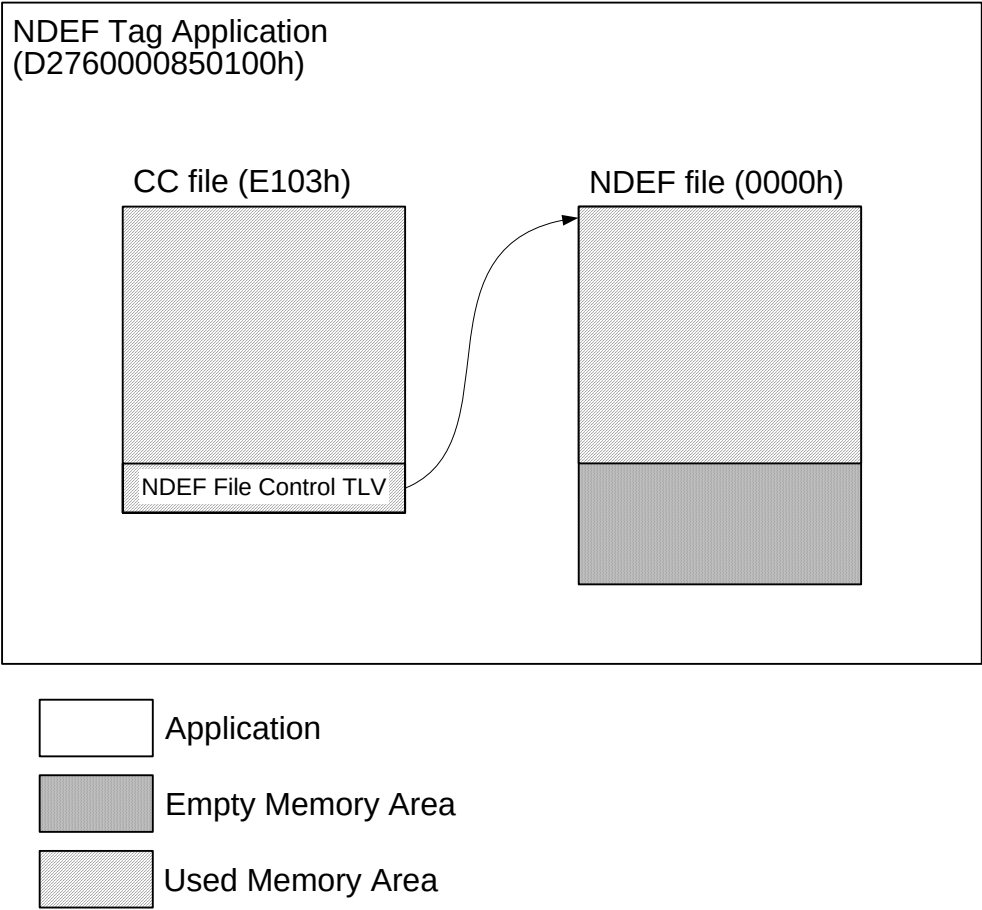


Figure 3: NDEF Tag Application Example

The Capability Container (CC) file is described in detail in Table 24.

Table 24: CC File Example

Offset	Size	Value	Content
0h	2	000Fh	CCLen (15 bytes)
2h	1	10h	Mapping Version 1.0 (this specification)
3h	2	003Bh	MLe (49 bytes); Maximum R-APDU data size
5h	2	0034h	MLc (52 bytes); Maximum C-APDU data size
7h	1	04h	T field of the NDEF File Control TLV
	1	06h	L field of the NDEF File Control TLV
	6		V field of the NDEF File Control TLV:
		0000h	File Identifier
		0032h	Maximum NDEF size (50 bytes)
		00h	NDEF file read access condition, read access without any security
		00h	NDEF file write access condition, write access without any security

The NDEF file is described in detail in Table 25.

Table 25: NDEF File Example

Offset	Size	Value	Content
0h	2	0003h	NLEN, NDEF length 3 bytes
2h	3	D00000h	Empty NDEF message

C. Example Command Flow

This chapter provides some examples of the command flow in order to show how a typical interaction on an APDU level could be performed by the NFC Forum device. It is assumed that the Type 4 tag platform is configured in a proper way and contains a valid NDEF file. The examples do not cover any check of the NDEF file, and they are related to the NDEF Tag Application described in B.

The commands and the responses are written in hexadecimal form with a space between each byte without the “h” character at the end e.g. 30 F3 AB 9C. The left-most byte is the first byte to be sent, and the right-most byte is the last one to be sent. The special acronyms Byte5 and Byte14 are used to indicate a group of bytes with a specific meaning indicated later on in the description of the command or of the response.

C.1 Detection of NDEF message

The example in this section detects the NDEF message applying the NDEF detection procedure (see section 6.4.1).

The first command is the NDEF Tag Application select (see section 6.4.2).

Command: 00 A4 04 00 07 D2 76 00 00 85 01 00

- The meanings of the bytes are the following ones:
 - 00h Class byte (CLA),
 - A4h Instruction byte (INS) for Select Command,
 - 04h Parameter byte (P1), select by name,
 - 00h Parameter byte (P2), first or only occurrence,
 - 07h Lc field,
 - D2760000850100h NDEF Tag Application name.

Expected Response: 90 00

- The meanings of the bytes are the following ones:
 - 9000h Status bytes (SW1, SW2), command completed.

The second command is the Capability Container select (see section 6.4.3):

Command: 00 A4 00 00 02 E1 03

- The meanings of the bytes are the following ones:
 - 00h Class byte (CLA),
 - A4h Instruction byte (INS) for Select command,
 - 00h Parameter byte (P1), select by identifier,
 - 00h Parameter byte (P2), first or only occurrence,
 - 02h Lc field,
 - E103h file identifier of the CC file.

Expected Response: 9000

- The meanings of the bytes are the following ones:
 - 9000h Status bytes (SW1, SW2), command completed.

The third command is ReadBinary data from CC file (see section 6.4.4):

Command: 00 B0 00 00 0F

- The meanings of the bytes are the following ones:
 - 00h Class byte (CLA),
 - B0h Instruction byte (INS) for ReadBinary command,
 - 0000h Parameter byte (P1, P2), offset inside the CC file,
 - 0Fh Le field,

Expected Response: 00 0F 10 00 3B 00 34 04 06 00 00 00 32 00 00 90 00

- The meanings of the bytes are the following ones:
 - 00 0Fh CCLen length of the CC file,
 - 10h Mapping Version 1.0 (this specification),
 - 003Bh MLe maximum 59 bytes R-APDU data size,
 - 0034h MLc maximum 52 bytes C-APDU data size,
 - NDEF File Control TLV
 - 04h T field of the NDEF File Control TLV,
 - 06h L field of the NDEF File Control TLV,
 - V field of the NDEF File Control TLV,
 - 0000h File Identifier of NDEF file,
 - 0032h Maximum NDEF file size of 50 bytes,
 - 00h read access without any security,
 - 00h write access without any security.

From the response it is possible to understand that the NDEF file identifier is 0000h, and the NDEF file has granted read access without any security. This allows the following two operations: the selection of the NDEF file and the read of the NDEF message.

The fourth command is the NDEF Select command (see section 6.4.5):

Command: 00 A4 00 00 02 00 00

- The meanings of the bytes are the following ones:
 - 00h Class byte (CLA),
 - A4h Instruction byte (INS) for Select command,
 - 00h Parameter byte (P1), select by identifier,
 - 00h Parameter byte (P2), first or only occurrence,
 - 02h Lc field,

- 0000h file identifier of the NDEF file retrieved from the CC file.

Expected Response: 90 00

- The meanings of the bytes are the following ones:
 - 9000h Status bytes (SW1, SW2), command completed.

The fifth command is the Read Binary command. The command reads the NLEN field of the NDEF file:

Command: 00 B0 00 00 02

- The meanings of the bytes are the following ones:
 - 00h Class byte (CLA),
 - B0h Instruction byte (INS) for ReadBinary command,
 - 0000h Parameter byte (P1, P2), offset inside the CC file,
 - 02h Le field,

Expected Response: 00 0F 90 00

- The meanings of the bytes are the following ones:
 - 000Fh NLEN length of the NDEF message,
 - 9000h Status bytes (SW1, SW2), command completed.

NLEN is smaller than the Maximum NDEF file size-2 (equal to $50-2=48$ bytes), and bigger than 0000h, hence the NDEF message is successfully detected inside the NDEF file.

C.2 Read data from the NDEF file

To read the NDEF file the NDEF read procedure is applied (see section 6.4.6). It is supposed that the NDEF file was previously successfully detected, and the NDEF file is correctly selected.

The command ReadBinary reads 15 bytes from the NDEF file:

Command: 00 B0 00 00 0F

- The meanings of the bytes are the following ones:
 - 00h Class byte (CLA),
 - B0h Instruction byte (INS) for ReadBinary command,
 - 0000h Parameter byte (P1, P2), offset inside the CC file,
 - 0Fh Le field,

Expected Response: 00 03 D0 00 00 Byte5...Byte14 90 00

- The meanings of the bytes are the following ones:
 - 0003h NLEN length of the NDEF message,
 - D00000h NDEF message field, it contains an empty NDEF message (see section Appendix A).
 - Byte5...Byte14 data that belong to the NDEF file that is ignored.
 - 9000h Status bytes (SW1, SW2), command completed.

C.3 Write data in the NDEF file

To write the NDEF file the NDEF update procedure is applied (see section 6.4.7). It is supposed that:

- The NDEF file was previously successfully detected (using the procedure described in section 6.4.1),
- The NDEF file has write access without any security granted,
- The NDEF file is correctly selected,
- The NDEF file size -2 (see Maximum NDEF file field of the CC file) is bigger than the NDEF message that has to be written into the NDEF file. In this example the NDEF message is 3 bytes long and the Maximum NDEF file = 50 bytes, hence $50-2 \geq 3$ it is allowed to write the NDEF message in the NDEF file.

NDEF Update command to write data into the NDEF file:

Command: 00 D6 00 00 05 00 03 D0 00 00

- The meanings of the bytes are the following ones:
 - 00h Class byte (CLA),
 - D6h Instruction byte (INS) for UpdateBinary command,
 - 00 00h Parameter byte (P1, P2), offset inside the CC file,
 - 05h Lc field,
 - 0003h NLEN, length of the NDEF message.
 - D00000h NDEF message composed of one empty record.

Expected Response: 9000

- The meanings of the bytes are the following ones:
 - 9000h Status bytes (SW1, SW2), command completed. Five data bytes have been successfully written to the NDEF file stating at the offset 0000h.

The writing of the NLEN and NDEF message field of the NDEF file has been done using one single NDEF Update command because the NLEN field and the NDEF message field are small enough (2 bytes + 3 bytes = 5 bytes) to be contained in the data field of the NDEF Update command (MLE=59 bytes).

D. Revision History

The following table outlines the revision history of Type 4 Tag Operation.

Table 26: Revision History

Document Name	Revision and Release Date	Status	Change Notice	Supersedes
NFCForum-TS-Type-4-Tag_1.0	1.0, July 2007	Final		



Generic Control Record Type Definition

Technical Specification

NFC Forum™

GC-RTD 1.0

NFCForum-TS-GenericControlRTD_1.0

2008-03-07

RESTRICTIONS ON USE

This specification is copyright © 2005-2008 by the NFC Forum, and was made available pursuant to a license agreement entered into between the recipient (Licensee) and NFC Forum, Inc. (Licensor) and may be used only by Licensee, and in compliance with the terms of that license agreement (License). If you are not the Licensee, you are not authorized to make any use of this specification. However, you may obtain a copy at the following page of Licensor's Website: http://www.nfc-forum.org/resources/spec_license after entering into and agreeing to such license terms as Licensor is then requiring. On the date that this specification was downloaded by Licensee, those terms were as follows:

1. LICENSE GRANT.

Licensor hereby grants Licensee the right, without charge, to copy (for internal purposes only) and share the Specification with Licensee's members, employees and consultants (as appropriate). This license grant does not include the right to sublicense, modify or create derivative works based upon the Specification.

2. NO WARRANTIES.

THE SPECIFICATION IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY, COMPLETENESS AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL LICENSOR, ITS MEMBERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE SPECIFICATION.

3. THIRD PARTY RIGHTS.

Without limiting the generality of Section 2 above, LICENSOR ASSUMES NO RESPONSIBILITY TO COMPILE, CONFIRM, UPDATE OR MAKE PUBLIC ANY THIRD PARTY ASSERTIONS OF PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS THAT MIGHT NOW OR IN THE FUTURE BE INFRINGED BY AN IMPLEMENTATION OF THE SPECIFICATION IN ITS CURRENT, OR IN ANY FUTURE FORM. IF ANY SUCH RIGHTS ARE DESCRIBED ON THE SPECIFICATION, LICENSOR TAKES NO POSITION AS TO THE VALIDITY OR INVALIDITY OF SUCH ASSERTIONS, OR THAT ALL SUCH ASSERTIONS THAT HAVE OR MAY BE MADE ARE SO LISTED.

4. TERMINATION OF LICENSE.

In the event of a breach of this Agreement by Licensee or any of its employees or members, Licensor shall give Licensee written notice and an opportunity to cure. If the breach is not cured within thirty (30) days after written notice, or if the breach is of a nature that cannot be cured, then Licensor may immediately or thereafter terminate the licenses granted in this Agreement.

5. MISCELLANEOUS.

All notices required under this Agreement shall be in writing, and shall be deemed effective five days from deposit in the mails. Notices and correspondence to the NFC Forum address as it appears below. This Agreement shall be construed and interpreted under the internal laws of the United States and the Commonwealth of Massachusetts, without giving effect to its principles of conflict of law.

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

Contents

1	Overview	1
1.1	Objectives	1
1.2	Purpose	1
1.3	Mission Statement and Goals	1
1.4	Applicable Documents or References	1
1.5	Administration	2
1.6	Special Word Usage	2
1.7	Name and Logo Usage	2
1.8	Intellectual Property	3
1.9	Glossary	3
2	Generic Control	4
2.1	Introduction	4
2.2	Dependencies	4
3	NDEF Structure	5
3.1	Message Sequence	5
3.2	Record Mapping	5
3.2.1	Structure	5
3.2.2	Syntax	5
3.2.3	Sequence Control for Multiple Records	6
3.3	List of Records	6
3.3.1	The Target Record	6
3.3.2	The Action Record	6
3.3.3	The Data Record	7
4	Implementation Considerations	8
4.1	Record Handling Architecture	8
4.2	Remarks on Security and Privacy	8
4.3	Authentication	8
4.4	Response from the Destination Device to Source Device	9
A.	Examples	10
A.1	A Generic Control Record to Access an Application	10
A.2	A Generic Control Record to Access a Property	11
A.3	Multiple Generic Control Records on a Tag	13
A.4	Using AT Commands for Mobile Terminals	17
B.	Test Requirements	19
C.	Revision History	20

Figures

Figure 1: The Structure of a Generic Control Record	5
Figure 2: Record Handling Architecture	8

Tables

Table 1: Configuration Byte of a Generic Control Record	6
Table 2: Action Flag Byte	7
Table 3: Action Numeric Codes	7
Table 4: Example of a Generic Control RTD Message	10
Table 5: Example of a Generic Control RTD Message to Set a Device Property	12
Table 6: Example of a Message Containing Multiple Generic Control Records	13
Table 7: Using AT Commands Example	17
Table 8: Revision History	20

1 Overview

The Generic Control Record Type Definition defines an NFC Forum Well Known Type on how to activate a specific application or to set a property in a destination device through NFC communication.

1.1 Objectives

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

1.2 Purpose

The purpose of the Generic Control RTD is to provide a way to request a specific action to an NFC Forum device (a destination device) from another NFC Forum device, a tag, or a card (source device) through NFC communication. A MIME type record in an NDEF message can provide access to an associated function; however, the association is defined solely in the destination device. It implies that a function to be accessed may differ from device to device when more than one function shares the same MIME type. To prevent such uncertainty, the Generic Control RTD allows the message issuer to specify specific functions to be accessed with the message. Moreover, the NDEF parser does not need to resolve the association of data and functions.

1.3 Mission Statement and Goals

To define a simple and generic RTD for manipulating (activating and operating) various applications and functions in smart equipment with NFC capabilities, such as a mobile phone, a PDA, and vice versa through NFC so that the generic RTD:

- Avoids a flood of unnecessary external record types for virtually identical purposes
- Saves resources (storage and processing) on NFC Forum devices (NFC-enabled equipment)
- Helps application providers to avoid defining new external record types, which increases complexity, resources, and management

1.4 Applicable Documents or References

Document Name	Document Citation
[JPEG]	E. Hamilton: "JPEG File Interchange Format (version 1.02)". September 1, 1992. http://www.w3.org/Graphics/JPEG/
[NDEF]	"NFC Data Exchange Format Specification", NFC Forum, 2006.
[NFC RTD]	"NFC Record Type Definition (RTD) Specification", NFC Forum, January 2006.
[PNG]	"Portable Network Graphics (PNG) Specification (Second Edition)", W3C Recommendation 10 November 2003. http://www.w3.org/TR/PNG/
[RFC 2119]	S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Harvard University, March 1997.

[RFC 2046]	N. Freed, N. Borenstein, “Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types”. RFC 2046, Innosoft, First Virtual, November 1996.
[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.
[SP RTD]	“Smart Poster Record Type Definition”, NFC Forum, 2006.
[TEXT RTD]	“NFC Text Record Type Definition”, NFC Forum, 2006.
[URI RTD]	“URI Record Type Definition”, NFC Forum, 2006.
[URI SCHEME]	List of Uniform Resource Identifier (URI) schemes registered by IANA is available at: http://www.iana.org/assignments/uri-schemes
[3GPP AT]	“AT command set for user Equipment (UE)”, 3GPP TS 27.007, 3rd Generation Partnership Project.

1.5 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: +1 781-610-9864

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

The Reference Applications Framework technical working group maintains this specification.

1.6 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.

1.7 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 colour 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.

1.8 Intellectual Property

The Generic Control 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, and revised on June 23, 2007.

1.9 Glossary

HO

(Negotiated) Handover

NDEF

NFC Data Exchange Format

RTD

Record Type Definition

SP

Smart Poster

RFU

Reserved for Future Use

2 Generic Control

2.1 Introduction

A Generic Control RTD provides a simple way to virtually request any specific action to an NFC Forum device (a destination device) from another NFC Forum device, tag, or card (source device) through NFC communication.

It is designed with the philosophy to allow:

- Access to functions or applications which are not covered by MIME type records or other NFC Forum Global Type records.
- The source device to explicitly indicate a certain function or a certain application on the destination device to perform a certain action.

A MIME type record in an NDEF message provides indirect access to an associated function or an application. When there are multiple functionalities and/or applications on a destination device and some of them share the same MIME type, only one of them is chosen for processing data contained in the record. Association between the data and function or application is determined by the destination device. This may lead to a different result than the source device expects.

Other NFC Forum Global Type records implicitly assume a dedicated application. This does not resolve all the limitations of MIME type records because only a limited number of popular types are defined as NFC Forum Global Types.

To resolve the issue above, a Generic Control is capable of requesting any application on the destination device. It allows the Generic Control to explicitly specify a certain function or a certain application to be accessed.

2.2 Dependencies

If an NDEF message contains one or more records other than Generic Control records, the processing sequence of multiple records is undefined.

The first record MAY restrict the processing sequence of the later records and MAY set the context of processing as described in NFC Record Type Definition Technical Specification [NFC RTD].

3 NDEF Structure

3.1 Message Sequence

There is no specific message sequence as the information is just read and acted upon. The receiving device is not assumed to reply to the source device for the Generic Control RTD message. Another NFC communication can be started when the activated function is designed to do so.

3.2 Record Mapping

3.2.1 Structure

The content of a Generic Control payload is an NDEF message.

The content of this message consists of a Configuration Byte and several NDEF sub-records. A Generic Control record contains the following components.

- A *Configuration Byte* for specifying several profiles to help determine how to handle the rest of the payload
- A *Target record* for identifying a function to handle the data contained in the Generic Control record
- An *Action record* for specifying the desired action for the function identified by the Target record to be applied to the data
- A *Data record* for specifying data to be processed by the function identified by the Target record

The structure of a Generic Control record is shown in Figure 1. Type names and content of payloads are introduced in 3.2.2 for the top-level record and in 3.3 for sub-records.

"Gc"	Config Byte	"t"	Target Identifier	"a"	Action Flag Byte	Action Identifier	"d"	Data
		Type Name	Payload	Type Name	Payload		Type Name	Payload
Type Name	Payload							

Figure 1: The Structure of a Generic Control Record

3.2.2 Syntax

The Well-known Type for the Generic Control record type is "Gc" (in NFC binary encoding: 0x47, 0x63).

The first byte of the payload of a "Gc" record is its Configuration Byte. The meanings of each bit in the Configuration Byte are shown in Table 1. The rest of the payload contains sub-records of a Target record, an Action record, and a Data record. The Generic Control RTD does not assume any particular order for these sub-records inside the Generic Control payload. However, it is **RECOMMENDED** that a Target record is specified first, an Action record is specified next, and a Data record is specified at the end for ease of readability and efficiency of processing.

Table 1: Configuration Byte of a Generic Control Record

Bit	Name	Symbol	Explanation
0	RFU	-	MUST be zero (0).
1	Sequence Control	SC	Set (1) if Exit Condition is to be checked.
2		EC	Set (1) if the following records are to be ignored when this record is not processed successfully.
3...7	RFU	-	MUST be zero (0).

3.2.3 Sequence Control for Multiple Records

When multiple Generic Control records are contained in a top-level NDEF message, it is RECOMMENDED that the assumption is that those records are expected to be processed in FIFO (first-in, first-out) manner. Only a Generic Control record MUST appear in the top-level NDEF message.

Sequence Control Flags in the Configuration Byte provide criteria to terminate the process in the middle of multiple Generic Control record sequences, depending on the result of each Generic Control record.

The reading device MUST refer the Configuration Byte and follow the criteria.

3.3 List of Records

The following record types are defined as NFC Local Types. The scopes of the records are limited locally in a Generic Control record only.

3.3.1 The Target Record

The NFC Local Type Name of the Target record is “t” (0x74).

A Generic Control record MUST contain one and only one Target record.

A Target record contains an instance of a Text RTD record [TEXT RTD] or an URI RTD record [URI RTD]. The destination device SHALL be responsible for translation of the sub-record’s content. If the destination device does not understand the sub-record’s content, the destination device SHOULD simply ignore the Generic Control record that contains the unresolved Target record.

3.3.2 The Action Record

The Action record is a Local Type record specific to the Generic Control RTD, which specifies the requested action for the target function to handle the data.

The NFC Local Type Name of the Action record is “a” (0x61).

A Generic Control record MAY contain one Action record. Generic Control records MUST NOT contain more than one Action record. When the Action record is omitted, the default action of the function may be applied. The default action is up to each function.

If none of the entities identified with the Target record or the destination device understand the Action record content, the Generic Control record SHOULD simply be ignored. Interpretations, as actual actions, of content in an Action record are up to the target function.

The payload of an Action record contains one Action Flag Byte and one data record. The Action Flag Byte indicates the data record type. The Action Flag Byte **MUST** be the first byte in the payload. Table 2 shows details of the Action Flag Byte.

Table 2: Action Flag Byte

Bit	Name	Symbol	Explanation
0	Numeric Code	NC	Set (1) if a numeric code is used to specify the action.
1..7	RFU	-	MUST be zero (0).

When Bit 0 of the Action Flag Byte is clear (NC=0), then the rest of the payload contains a sufficient NDEF record such as a Text RTD record, MIME type record, or vice versa. The target function **SHOULD** be able to interpret the data to perform intended action.

When Bit 0 of the Action Flag Byte is set (NC=1), the next single byte is the numeric code of an action. The value of the single byte **SHALL** be interpreted as shown in Table 3.

Table 3: Action Numeric Codes

Value	Action
0	The default action of the target function (for example, opening a page for a Web browser, dialing a number for a phone, and so on).
1	Store for later use (for example, add a URL to bookmarks for a Web browser, store data as a phone number into a directory for a phone, and so on).
2	Open for editing (for example, open a URI in a URI editor for a Web browser, open a directory editor for a phone, and so on).
3..FF	RFU

The numeric encoding is only effective when the target function is capable of handling the codes as stated in the table. If an incapable function received one of the numeric codes above, it **MAY** ignore the code and perform the default action, or it **MAY** ignore the parent Generic Control record.

If a receiving device encounters a value marked as RFU, it **MAY** treat it as having a value of zero.

The numeric encoding is allowed for consistent expression with the Action Record of Smart Poster. Please refer Technical Specification of Smart Poster RTD [SP RTD] for more information.

3.3.3 The Data Record

The NFC Local Type Name of the Data record is “d” (0x64).

A Data record **MAY** contain any type of data. The data of records contained in the Data record **SHOULD** simply be passed to the target function.

Interpretations of content in a data record are up to the target function.

4 Implementation Considerations

4.1 Record Handling Architecture

To reside with processes to handle MIME type records and other NFC Global Type records, the architecture diagram concerning Generic Control records handling SHOULD be like that shown in Figure 2.

Once a record type is identified by the NDEF parser, the record data is forwarded to the record dispatcher and the dispatcher determines which handler will take care of the record. After the association between the data and a corresponding application (or function) is resolved by a handler (MIME handler for MIME type records and RTD handler for NFC Forum Global Type records), the application launcher launches the associated application, if necessary, and delivers the data to the application.

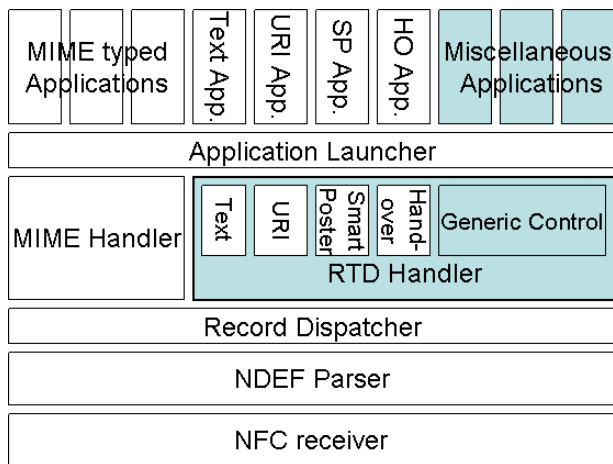


Figure 2: Record Handling Architecture

4.2 Remarks on Security and Privacy

The destination device is responsible for security issues that may be caused by execution of actions described in Generic Control records (for example, the destination device may produce a notification to its user).

The destination device is responsible for privacy issues that may be caused by execution of actions described in Generic Control records.

The sending device or issuer is responsible for privacy issues that may be caused by transferring data within Generic Control records. A Generic Control record simply conveys data with no specific consideration on privacy issues.

4.3 Authentication

A Generic Control record does not provide any specific feature for authenticating a partner device or an application. If authentication is required, it should be done before or after transferring a Generic Control record. Authentication is based on implementation.

When the destination device is in Read (/Write) mode and the source device is a tag or a card, authentication information may be contained in the data sub-record of the Generic Control record

that is stored in the tag or the card, so that the destination device or application can verify the authentication information before performing any action.

When two devices are communicating in Peer-to-Peer mode, authentication information may be exchanged before transferring a Generic Control record.

It is also possible to use a Generic Control record to convey authentication information for another Generic Control record which requires authenticated status. Association of these multiple Generic Control records should be managed by the communicating devices. Since a Generic Control record does not provide a feature for such association, some information to ensure the association should be embedded in a data sub-record of a Generic Control record.

4.4 Response from the Destination Device to Source Device

A Generic Control record RTD basically describes a record of a one-way transfer. When a response from the destination device to the source device is required, it is possible to use another Generic Control record (which is transferred in the opposite direction by swapping the role of the source device and the destination device) as a response.

When such mutual communication is expected, maintaining consistency of protocol among two devices is the responsibility of the devices' applications.

It is recommended that the source device should be robust enough that it can recover properly, even when the destination device does not send back any response.

A. Examples

The contents of this appendix are informative.

The following examples present a full NDEF message, as if they were read from a NFC Forum tag or received from another NFC Forum device.

A.1 A Generic Control Record to Access an Application

The table below is an example of the Generic Control record which will add 500 points to a customer royalty bonus application. The target application is identified as “file://localhost/Appli/CustomerBonus” with URI style notation. The Action “add” and the data “500” are specified as Text records.

Table 4: Example of a Generic Control RTD Message

Offset	Content	Length	Explanation
0	0xD1	1	NDEF header. TNF=0x01 (Well Known Type), SR=1, MB=1, ME=1
1	0x02	1	Record Name Length
2	0x4A	1	Length of the Generic Control data (74Bytes)
3	“Gc”	2	The record name
5	0x00	1	Configuration Byte SC=0, EC=0
6	0xD1	1	NDEF header. TNF=0x00, SR=1, MB=1, ME=1
7	0x01	1	Record Name Length
8	0x22	1	Length of the Target data (34 Bytes)
9	“t”	1	The record name of Target record
10	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
11	0x01	1	Record Name Length
12	0x1E	1	Length of the Target data (30 Bytes)
13	“U”	1	The record name
14	0x1D	1	URI identifier code (“file://”)
15	“localhost/Appli/CustomerBonus”	29	Value of Target
44	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
45	0x01	1	Record Name Length
46	0x0E	1	Length of the Target data (14 Bytes)

Offset	Content	Length	Explanation
47	“a”	1	The record name of Action record
48	0x00	1	Action Flag NC = 0 (text name)
49	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
50	0x01	1	Record Name Length
51	0x09	1	Length of the Target data (9 Bytes)
52	“T”	1	The record name
53	0x05	1	Status Byte
54	“en-US”	5	Language Code
59	“add”	3	Value of Action
62	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
63	0x01	1	Record Name Length
64	0x0D	1	Length of the Target data (13 Bytes)
65	“d”	1	The record name of Data record
66	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
67	0x01	1	Record Name Length
68	0x09	1	Length of the Target data (9 Bytes)
69	“T”	1	The record name
70	0x05	1	Status Byte
71	“en-US”	5	Language Code
76	“500”	3	Value of Data

A.2 A Generic Control Record to Access a Property

The table below is an example of the Generic Control record which sets a mobile phone to silent mode. This example assumes that the function to manage the properties such as Silent Mode, Ringer Tone, Language Preference, and so on is identified by “PropertyManager”. It also assumes that the Property Manager accepts a command “Set”, a parameter name string “SilentMode”, and data string “On” to activate Silent Mode.

Table 5: Example of a Generic Control RTD Message to Set a Device Property

Offset	Content	Length	Explanation
0	0xD1	1	NDEF header. TNF=0x01 (Well Known Type), SR=1, MB=1, ME=1
1	0x02	1	Record Name Length
2	0x54	1	Length of the Generic Control data (84Bytes)
3	“Gc”	2	The record name
5	0x00	1	Configuration Byte SC=0, EC=0
6	0xD1	1	NDEF header. TNF=0x00, SR=1, MB=1, ME=1
7	0x01	1	Record Name Length
8	0x19	1	Length of the Generic Control data (25Bytes)
9	“t”	1	The record name of Target record
10	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
11	0x01	1	Record Name Length
12	0x15	1	Length of the Target data (21 Bytes)
13	“T”	1	The record name
14	0x05	1	Status Byte
15	“en-US”	5	Language Code
20	“PropertyManager”	15	Value of Target
35	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
36	0x01	1	Record Name Length
37	0x0E	1	Length of the Target data (14 Bytes)
38	“a”	1	The record name of Action record
39	0x00	1	Action Flag NC = 0 (text name)
40	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
41	0x01	1	Record Name Length
42	0x09	1	Length of the Target data (9 Bytes)
43	“T”	1	The record name
44	0x05	1	Status Byte

Offset	Content	Length	Explanation
45	“en-US”	5	Language Code
50	“Set”	3	Value of Action
53	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
54	0x01	1	Record Name Length
55	0x20	1	Length of the Target data (32 Bytes)
56	“d”	1	The record name of Data record
57	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
58	0x01	1	Record Name Length
59	0x10	1	Length of the Target data (16 Bytes)
60	“T”	1	The record name
61	0x05	1	Status Byte
62	“en-US”	5	Language Code
67	“SilentMode”	10	Value of Data
77	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
78	0x01	1	Record Name Length
79	0x08	1	Length of the Target data (8 Bytes)
80	“T”	1	The record name
81	0x05	1	Status Byte
82	“en-US”	5	Language Code
87	“ON”	2	Value of Data

A.3 Multiple Generic Control Records on a Tag

Below is an example of Multiple Generic Control records. The first record sets the Silent Mode of the device to OFF. The second record accesses the Music Player and plays (music) data reachable with the specified URI. If the first record causes an error, the second record MAY be abandoned.

Table 6: Example of a Message Containing Multiple Generic Control Records

Offset	Content	Length	Explanation
0	0xD1	1	NDEF header. TNF=0x01 (Well Known Type), SR=1, MB=1, ME=1
1	0x02	1	Record Name Length

Offset	Content	Length	Explanation
2	0x55	1	Length of the Generic Control data (85Bytes)
3	“Gc”	2	The record name
5	0x06	1	Configuration Byte SC=1, EC=1
6	0xD1	1	NDEF header. TNF=0x00, SR=1, MB=1, ME=1
7	0x01	1	Record Name Length
8	0x19	1	Length of the Generic Control data (25Bytes)
9	“t”	1	The record name of Target
10	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
11	0x01	1	Record Name Length
12	0x15	1	Length of the Target data (21 Bytes)
13	“T”	1	The record name
14	0x05	1	Status Byte
15	“en-US”	5	Language Code
20	“PropertyManager”	15	Value of Target
35	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
36	0x01	1	Record Name Length
37	0x0E	1	Length of the Generic Control data (14Bytes)
38	“a”	1	The record name of Action record
39	0x00	1	Action Flag NC = 0 (text name)
40	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
41	0x01	1	Record Name Length
42	0x09	1	Length of the Target data (9 Bytes)
43	“T”	1	The record name
44	0x05	1	Status Byte
45	“en-US”	5	Language Code
50	“Set”	3	Value of Action

Offset	Content	Length	Explanation
53	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
54	0x01	1	Record Name Length
55	0x21	1	Length of the Generic Control data (33Bytes)
56	“d”	1	The record name of Data record
57	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
58	0x01	1	Record Name Length
59	0x10	1	Length of the Target data (16 Bytes)
60	“T”	1	The record name
61	0x05	1	Status Byte
62	“en-US”	5	Language Code
67	“SilentMode”	10	Value of Data
77	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
78	0x01	1	Record Name Length
79	0x09	1	Length of the Target data (9 Bytes)
80	“T”	1	The record name
81	0x05	1	Status Byte
82	“en-US”	5	Language Code
87	“OFF”	3	Value of Data
90	0xD1	1	NDEF header. TNF=0x01 (Well Known Type), SR=1, MB=1, ME=1
91	0x02	1	Record Name Length
92	0x54	1	Length of the Generic Control data (84Bytes)
93	“Gc”	2	The record name
95	0x00	1	Configuration Byte SC=0, EC=0
96	0xD1	1	NDEF header. TNF=0x00, SR=1, MB=1, ME=1
97	0x01	1	Record Name Length
98	0x20	1	Length of the Target data (32Bytes)

Offset	Content	Length	Explanation
99	"t"	1	The record name of Target record
100	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
101	0x01	1	Record Name Length
102	0x1C	1	Length of the Target data (28 Bytes)
103	"U"	1	The record name
104	0x1D	1	URI identifier code ("file://")
105	"localhost/Appli/MusicPlayer"	27	Value of Target
132	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
133	0x01	1	Record Name Length
134	0x0F	1	Length of the Target data (15Bytes)
135	"a"	1	The record name of Action record
136	0x00	1	Action Flag NC = 0 (text name)
137	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
138	0x01	1	Record Name Length
139	0x0A	1	Length of the Target data (10 Bytes)
140	"T"	1	The record name
141	0x05	1	Status Byte
142	"en-US"	5	Language Code
147	"play"	4	Value of Action
151	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
152	0x01	1	Record Name Length
153	0x18	1	Length of the Target data (24Bytes)
154	"d"	1	The record name of Data record
155	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
156	0x01	1	Record Name Length
157	0x14	1	Length of the Target data (20 Bytes)
158	"U"	1	The record name
159	0x03	1	URI identifier code ("http://")

Offset	Content	Length	Explanation
160	"nfc-forum.org/music"	19	Value of Data

A.4 Using AT Commands for Mobile Terminals

AT command sets [3GPP AT] are widely adopted for mobile handset terminals to control their functions. Using AT commands through NFC helps to control the destination device.

This example shows a message to set the "Vibrator mode" to "ON" with an AT command. In this example, Terminal Adapter (which is a component of a Mobile Terminal) to accept AT commands is assumed to be identified as "file://localhost/TA".

Many AT commands expect a response from Terminal Adapter after receiving a command. Please see 4.4 when handling such transactions.

Table 7: Using AT Commands Example

Offset	Content	Length	Explanation
0	0xD1	1	NDEF header. TNF=0x01 (Well Known Type), SR=1, MB=1, ME=1
1	0x02	1	Record Name Length
2	0x31	1	Length of the Generic Control data (49Bytes)
3	"Gc"	2	The record name
5	0x00	1	Configuration Byte SC=0, EC=0
6	0xD1	1	NDEF header. TNF=0x00, SR=1, MB=1, ME=1
7	0x01	1	Record Name Length
8	0x11	1	Length of the Generic Control data (17Bytes)
9	"t"	1	The record name of Target record
10	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
11	0x01	1	Record Type Name Length
12	0x0D	1	Length of the Target data (13 Bytes)
13	"U"	1	The record name
14	0x1D	1	Abbreviation of "file://"
15	"localhost/TA"	12	Value of Target
27	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
28	0x01	1	Record Name Length
29	0x02	1	Length of the Generic Control data (2Bytes)

Offset	Content	Length	Explanation
30	“a”	1	The record name of Action record
31	0x01	1	Action Flag NC = 1 (numeric code)
32	0x00	1	Value of Action (the default action)
33	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
34	0x01	1	Record Name Length
35	0x11	1	Length of the Generic Control data (17Bytes)
36	“d”	1	The record name of Data record
37	0xD1	1	NDEF header. TNF=0x01, SR=1, MB=1, ME=1
38	0x01	1	Record Name Length
39	0x0D	1	Length of the Generic Control data (13Bytes)
40	“T”	1	The record name
41	0x05	1	Status Byte
42	“en-US”	5	Language Code
47	“+CVIB=1”	7	Value of Data

B. Test Requirements

It is recommended that the following requirements are satisfied when an NFC-equipped device is capable of handling (producing/parsing) NDEF records with the Generic Control type, although the capability itself is *Optional* for an NFC Forum-compliant device.

- TR-RTD-GC-1: A Generic Control record **MUST** be in the form of an NDEF record with NFC Forum Well-Known Type “Gc” (in NFC binary encoding: 0x47, 0x63).
- TR-RTD-GC-2: A Generic Control record **MUST** contain a Configuration Byte (shown in Table 1) for the first byte of the payload.
- TR-RTD-GC-3: A Generic Control record **MUST** contain one and only one target (NFC Forum Local Type) record as a sub-record.
- TR-RTD-GC-4: A Generic Control record **MAY** contain one and only one action (NFC Forum Local Type) record as a sub-record.
- TR-RTD-GC-5: A Generic Control record **MAY** contain one and only one data (NFC Forum Local Type) record as a sub-record.
- TR-RTD-GC-6: Type name of each sub-records (target, action and data) **MUST** be stated as “t”, “a” and “d” respectively.
- TR-RTD-GC-7: A target (NFC Forum Local Type) record **MUST** contain a URI record or Text record in its payload.
- TR-RTD-GC-8: An action (NFC Forum Local Type) record **MUST** contain an Action Flag Byte (shown in Table 2) at the first byte of its payload followed by an Action Identifier (defined in 3.3.2) when it is present.
- TR-RTD-GC-9: The Action Identifier **MUST** be in the numeric form when the NC Flag in the corresponding Action Flag Byte is set (1).
- TR-RTD-GC-10: A data (NFC Forum Local Type) record **MUST** contain one or more sub-records of any type in its payload when it is present.
- TR-RTD-GC-11: Multiple Generic Control records **MAY** appear in an NDEF message containing multiple NDEF records.
- TR-RTD-GC-12: Only Generic Control records **MUST** appear in an NDEF message starting with a Generic Control record.

C. Revision History

The following table outlines the revision history of Generic Control Record Type Definition.

Table 8: Revision History

Document Name	Revision and Release Date	Status	Change Notice	Supersedes
Generic Control Record Type Definition	1.0, March 2008	Final	None	



NFC Record Type Definition (RTD)

Technical Specification

NFC ForumTM

RTD 1.0

NFCForum-TS-RTD_1.0

2006-07-24

RESTRICTIONS ON USE

This specification is copyright © 2005-2006 by the NFC Forum, and was made available pursuant to a license agreement entered into between the recipient (Licensee) and NFC Forum, Inc. (Licensor) and may be used only by Licensee, and in compliance with the terms of that license agreement (License). If you are not the Licensee, you are not authorized to make any use of this specification. However, you may obtain a copy at the following page of Licensor's Website: http://www.nfc-forum.org/resources/spec_license after entering into and agreeing to such license terms as Licensor is then requiring. On the date that this specification was downloaded by Licensee, those terms were as follows:

1. LICENSE GRANT.

Licensor hereby grants Licensee the right, without charge, to copy (for internal purposes only) and share the Specification with Licensee's members, employees and consultants (as appropriate). This license grant does not include the right to sublicense, modify or create derivative works based upon the Specification.

2. NO WARRANTIES.

THE SPECIFICATION IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY, COMPLETENESS AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL LICENSOR, ITS MEMBERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE SPECIFICATION.

3. THIRD PARTY RIGHTS.

Without limiting the generality of Section 2 above, LICENSOR ASSUMES NO RESPONSIBILITY TO COMPILE, CONFIRM, UPDATE OR MAKE PUBLIC ANY THIRD PARTY ASSERTIONS OF PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS THAT MIGHT NOW OR IN THE FUTURE BE INFRINGED BY AN IMPLEMENTATION OF THE SPECIFICATION IN ITS CURRENT, OR IN ANY FUTURE FORM. IF ANY SUCH RIGHTS ARE DESCRIBED ON THE SPECIFICATION, LICENSOR TAKES NO POSITION AS TO THE VALIDITY OR INVALIDITY OF SUCH ASSERTIONS, OR THAT ALL SUCH ASSERTIONS THAT HAVE OR MAY BE MADE ARE SO LISTED.

4. TERMINATION OF LICENSE.

In the event of a breach of this Agreement by Licensee or any of its employees or members, Licensor shall give Licensee written notice and an opportunity to cure. If the breach is not cured within thirty (30) days after written notice, or if the breach is of a nature that cannot be cured, then Licensor may immediately or thereafter terminate the licenses granted in this Agreement.

5. MISCELLANEOUS.

All notices required under this Agreement shall be in writing, and shall be deemed effective five days from deposit in the mails. Notices and correspondence to the NFC Forum address as it appears below. This Agreement shall be construed and interpreted under the internal laws of the United States and the Commonwealth of Massachusetts, without giving effect to its principles of conflict of law.

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

Contents

1	Introduction.....	1
1.1	Objectives.....	1
1.2	Purpose	1
1.2.1	Mission Statement and Goals.....	1
1.3	References	2
1.4	Administration.....	2
1.5	Special Word Usage	2
1.6	Name and Logo Usage	3
1.7	Intellectual Property	3
1.8	Glossary.....	3
2	Record Types.....	5
2.1	NFC Forum Well-known Type	5
2.1.1	NFC Forum Global Type	5
2.1.2	NFC Forum Local Type	6
2.2	NFC Forum External Type.....	6
2.3	Record Types Generic Requirements.....	7
3	RTD Type Names	8
3.1	Binary Encoding.....	9
3.2	Percent Encoding in NFC Forum Types	9
3.3	Equivalence of Record Type Names	9
3.4	RTD Type Names Requirements.....	10
4	Error Handling	11
4.1	Illegal characters.....	11
4.2	Unknown Record Types.....	11
4.3	Error Handling Requirements	11
A.	Character Set for Record Types.....	12
B.	Record Type Name Examples	13
C.	Discussion on Associating Records	14
D.	Revision History	16

Figures

Figure 1.	NDEF Messages (Multiple)	14
Figure 2.	NDEF Message with Metadata	14

Tables

Table 1. Definitions	3
Table 2. Acronyms	4
Table 3. ASCII Character Chart	12
Table 4. Translating Record Type Names into Binary Representation	13
Table 5. Revision History	16

Test Requirements

Test Requirements 1. Record Types Generic Requirements	7
Test Requirements 2. RTD Type Names Requirements.....	10
Test Requirements 3. Error Handling Requirements.....	11

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 <i>NFC DATA EXCHANGE FORMAT (NDEF, Re-Draft Revision 0.96)</i> , 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"
 "f0oBaR"
 "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:f0oBaR"

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 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	



Smart Poster Record Type Definition

Technical Specification

NFC Forum™

SPR 1.1

NFCForum-SmartPoster_RTD_1.0

2006-07-24

RESTRICTIONS ON USE

This specification is copyright © 2005-2006 by the NFC Forum, and was made available pursuant to a license agreement entered into between the recipient (Licensee) and NFC Forum, Inc. (Licensor) and may be used only by Licensee, and in compliance with the terms of that license agreement (License). If you are not the Licensee, you are not authorized to make any use of this specification. However, you may obtain a copy at the following page of Licensor's Website: http://www.nfc-forum.org/resources/spec_license after entering into and agreeing to such license terms as Licensor is then requiring. On the date that this specification was downloaded by Licensee, those terms were as follows:

1. LICENSE GRANT.

Licensor hereby grants Licensee the right, without charge, to copy (for internal purposes only) and share the Specification with Licensee's members, employees and consultants (as appropriate). This license grant does not include the right to sublicense, modify or create derivative works based upon the Specification.

2. NO WARRANTIES.

THE SPECIFICATION IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY, COMPLETENESS AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL LICENSOR, ITS MEMBERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE SPECIFICATION.

3. THIRD PARTY RIGHTS.

Without limiting the generality of Section 2 above, LICENSOR ASSUMES NO RESPONSIBILITY TO COMPILE, CONFIRM, UPDATE OR MAKE PUBLIC ANY THIRD PARTY ASSERTIONS OF PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS THAT MIGHT NOW OR IN THE FUTURE BE INFRINGED BY AN IMPLEMENTATION OF THE SPECIFICATION IN ITS CURRENT, OR IN ANY FUTURE FORM. IF ANY SUCH RIGHTS ARE DESCRIBED ON THE SPECIFICATION, LICENSOR TAKES NO POSITION AS TO THE VALIDITY OR INVALIDITY OF SUCH ASSERTIONS, OR THAT ALL SUCH ASSERTIONS THAT HAVE OR MAY BE MADE ARE SO LISTED.

4. TERMINATION OF LICENSE.

In the event of a breach of this Agreement by Licensee or any of its employees or members, Licensor shall give Licensee written notice and an opportunity to cure. If the breach is not cured within thirty (30) days after written notice, or if the breach is of a nature that cannot be cured, then Licensor may immediately or thereafter terminate the licenses granted in this Agreement.

5. MISCELLANEOUS.

All notices required under this Agreement shall be in writing, and shall be deemed effective five days from deposit in the mails. Notices and correspondence to the NFC Forum address as it appears below. This Agreement shall be construed and interpreted under the internal laws of the United States and the Commonwealth of Massachusetts, without giving effect to its principles of conflict of law.

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

Contents

1 Overview	1
1.1 Objectives	1
1.2 Purpose	1
1.2.1 Mission Statement and Goals	1
1.3 Applicable Documents	1
1.4 Administration	2
1.5 Special Word Usage	2
1.6 Name and Logo Usage	2
1.7 Intellectual Property	3
1.8 Acronyms	3
2 Smart Poster	4
2.1 Introduction	4
3 NDEF Structure	5
3.1 Messaging Sequence	5
3.2 Records Mapping	5
3.2.1 Syntax	5
3.2.2 Structure	5
3.3 List of Records	6
3.3.1 The URI Record	6
3.3.2 The Title Record	6
3.3.3 The Recommended Action Record	6
3.3.4 The Icon Record	7
3.3.5 The Size Record	7
3.3.6 The Type Record	8
3.4 Dependencies	8
A. Examples	9
B. Revision History	12

Tables

Table 1. Acronyms	3
Table 2. Action Record Values	6
Table 3. The Size Record Layout	7
Table 4. Example for a Simple URI	9
Table 5. Example for a Complex URI	10
Table 6. Revision History	12

1 Overview

The Smart Poster Record Type Definition defines an NFC Forum Well-known Type on how to put URLs, SMSs, or phone numbers on an NFC Forum Tag or how to transport them between devices.

1.1 Objectives

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

1.2 Purpose

1.2.1 Mission Statement and Goals

The URI RTD specifies a way to mark different kinds of URIs and IRIs, but often you need a way to associate metadata with the URI. The purpose of the Smart Poster RTD is to provide the necessary wrapper to fulfill the Smart Poster use case, as defined by the NFC Forum.

In the Smart Poster use case, information about an object, event, etc, is somehow attached onto a physical object. The typical example is a movie poster which contains a tag with the Smart Poster record. When the user touches it with his NFC-enabled device (such as a cell phone or a PDA), a browser window opens and the device connects to the Internet to fetch that data. Another possibility might be that the device (if it's a cell phone) sends an SMS to a number defined on the tag to access some value-added service.

The design goal of the Smart Poster was to provide a simple way to access a remote service by using the touch paradigm.

1.3 Applicable Documents

[JPEG]	E. Hamilton: "JPEG File Interchange Format (version 1.02)". September 1, 1992. http://www.w3.org/Graphics/JPEG/
[NDEF]	"NFC Data Exchange Format Specification", NFC Forum, 2006.
[NFC RTD]	"NFC Record Type Definition (RTD) Specification", NFC Forum, 2006.
[PNG]	"Portable Network Graphics (PNG) Specification (Second Edition)". W3C Recommendation, 10 November 2003. http://www.w3.org/TR/PNG/
[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 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
[TEXT]	"NFC Text RTD Specification", NFC Forum, 2006.
[URI]	"URI RTD Specification", NFC Forum, 2006.

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

1.4 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.

[List of participating companies in working group goes in this section.]

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 Smart Poster 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 section defines all relevant 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
RTD	Record Type Description

2 Smart Poster

2.1 Introduction

The Smart Poster is one of the key use cases for NFC technology. The idea is that an object can be made “smart”, i.e., it is capable of storing additional information about itself in the form of an NFC Forum Tag. By touching an NFC Forum Device to the tag, this information can be read and displayed to the user.

The Smart Poster can also contain *actions* that will trigger an application in the device; for example, launching a browser to view a web site, or sending an SMS to a premium service to receive a ring tone.

The Smart Poster concept is built around URIs (Uniform Resource Identifiers [RFC 3986]), which have become the standard for referencing information around the Internet. URIs are very powerful, and they can represent anything from unique identifiers to EPC codes to web addresses to SMS messages to phone calls and beyond. The Smart Poster Record defines a superstructure that associates a URI with various types of metadata.

3 NDEF Structure

3.1 Messaging Sequence

There is no specific messaging sequence, as the information is just read and acted upon. The receiving device is not assumed to reply to the source device.

3.2 Records Mapping

3.2.1 Syntax

The content of a Smart Poster payload is an NDEF message. The contents of this message consist of several NDEF records. The Smart Poster can have zero, one, or more of the following components:

- **The Title record** for the service (there can be many of these in different languages, but a language **MUST NOT** be repeated). This record is optional.
- **The URI record**. This is the core of the Smart Poster, and all other records are just metadata about this record. There **MUST** be one URI record and there **MUST NOT** be more than one.
- **The Action record**. This record describes how the service should be treated. For example, the action may indicate that the device should save the URI as a bookmark or open a browser. The Action record is optional. If it does not exist, the device may decide what to do with the service. If the action record exists, it should be treated as a strong suggestion; the UI designer may ignore it, but doing so will induce a different user experience from device to device.
- **The Icon record**. A Smart Poster may include an icon by including one or many MIME-typed image records within the Smart Poster. If the device supports images, it **SHOULD** select and display one of these, depending on the device capabilities. The device **SHOULD** display only one. The Icon record is optional.
- **The Size record**. If the URI references an external entity (e.g., by URL), the Size record may be used to tell how large the object is. This is useful if the reader device needs to decide in advance whether it has the capability to process the referenced object. The Size record is optional.
- **The Type record**. If the URI references an external entity (e.g., via a URL), the Type record may be used to declare the MIME type of the entity. This can be used to tell the mobile device what kind of an object it can expect before it opens the connection. The Type record is optional.

There **MAY** be other records, which can be treated in an application-specific manner. For example, some application might include a vCard contact card using the proper MIME type. Applications **MAY** ignore any extra records inside the Smart Poster.

The Well-known Type for the Smart Poster record type is “Sp” (in NFC binary encoding: 0x53, 0x70)

3.2.2 Structure

The Smart Poster RTD does not assume any particular order for the NDEF records inside the master record.

3.3 List of Records

3.3.1 The URI Record

There is only one URI record per Smart Poster record. This is also the only mandatory record within a Smart Poster.

The device is not required to support any particular URI protocol, but if the device does not support the referenced protocol, it **MUST** discard the entire Smart Poster record.

3.3.2 The Title Record

The Title record is an instance of a Text RTD Record [TEXT]. There **MAY** be an arbitrary number of title records in the Smart Poster. However, there **MUST NOT** be two or more records with the same language identifier.

The Title record **SHOULD** be shown to the user.

NOTE TO IMPLEMENTERS: The implementer should be aware of the fact that by putting malicious information to the Title record and thus misrepresenting the service, it might be possible to fool the user into thinking that the tag contents might be something else entirely. This is a so-called *phishing* technique. For example, if the Title record contains the text “http://www.internetbanking.com”, and the URI record the text “http://myevilsite.com”, the user might be fooled into giving his banking information, if the Title record is the only one that is shown to the user.

3.3.3 The Recommended Action Record

The Action record is a Local Type specific to the Smart Poster. It suggests a course of action that the device should do with the content. The syntax is as follows:

The NFC Local Type Name for the action is “act” (0x61 0x63 0x74). The action record is defined as having a local scope only, and therefore it has meaning only within a Smart Poster record. A lone “act” –record **SHALL** be considered an error.

The content is a single byte, which **SHALL** be interpreted as follows:

Table 2. Action Record Values

Value	Action
0	Do the action (send the SMS, launch the browser, make the telephone call)
1	Save for later (store the SMS in INBOX, put the URI in a bookmark, save the telephone number in contacts)
2	Open for editing (open an SMS in the SMS editor, open the URI in an URI editor, open the telephone number for editing).
3..FF	RFU

The device **MAY** ignore this suggestion. The default (i.e., the Action record is missing from the Smart Poster) is not defined. For example, the device might show a list of options to the user.

If the device encounters a value marked as RFU, it MAY treat it as value zero, or ignore the entire “act” record.

3.3.4 The Icon Record

The Smart Poster may also contain a number of Icon records that have an image NDEF MIME type [NDEF], [RFC2046]. For example, “image/jpeg”, “image/png”, etc. There might also be animated Icon records, using common media types such as “video/mpeg”. A reader device SHOULD display one (and only one) of these Icons to the user prior to acting on the URI record.

A reader device is not required to support any particular image format, but it is RECOMMENDED that implementers stick to well-known, well-compressed formats such as PNG [PNG] and JPEG [JPEG].

3.3.5 The Size Record

The Size Record contains a four-byte, 32-bit, unsigned integer, which contains the size of object that the URI field refers to. Note that in practice this is limited to URLs (http://, ftp:// and similar).

The Size Record’s Local Type Name is “s”.

The size is expressed in network byte order (most significant byte first).

Table 3. The Size Record Layout

0	1	2	3
Byte 0	Byte 1	Byte 2	Byte 3

For example, if Byte 0 contains 0x12, Byte 1 contains 0x34, Byte 2 contains 0x56, and Byte 3 0x78, the size of the referred object is 0x12345678 bytes.

The size record MAY be used by the device to determine whether it can accommodate the referenced file or not. For example, an NFC tag could trigger the download of an application to a cell phone. Using a combination of the Type Record and the Size Record, the mobile phone could determine whether it can accommodate such a program or not.

The Size Record is for informational purposes only. Since the object size in the network may vary (for example, due to updates), this value should be used as a guideline only.

The Size Record is optional to support.

3.3.6 The Type Record

The Payload of the Type Record is a UTF-8–formatted string that describes a MIME type [RFC 2046] which describes the type of the object that can be reached through the URI. (In practice this is limited to URLs only, much like the Size Record.)

The Local Type Name for the Type Record is “t”.

The length of the payload string is the same as the length of the payload, so there is no need for separate length information or termination.

The Type Record MAY be used by the device to determine whether it can process the referenced file or not. For example, an NFC tag could trigger a media file playback from an URL. If the Type Record references an unknown media type, the reader device (e.g. a cell phone) does not need to even initiate the playback.

The Type Record is optional to support.

3.4 Dependencies

If an NDEF message contains one or multiple URI [URI] records in addition to the Smart Poster record at the top level (i.e., not nested), the Smart Poster record overrides them. The NDEF application MUST use only the Smart Poster record.

A. Examples

The following examples display a full NDEF message, as if it was read from a tag or received from another NFC device. The bit combination in the NDEF header may vary if you are embedding the Smart Poster in some other NDEF message.

A.1 Simple URI on a Tag

The contents of the record would look like this (total length 23 bytes).

Table 4. Example for a Simple URI

Offset	Content	Length	Explanation
0	0xD1	1	NDEF header. TNF = 0x01 (Well Known Type). SR=1, MB=1, ME=1
1	0x02	1	Record name length (2 bytes)
2	0x12	1	Length of the Smart Poster data (18 bytes)
3	“Sp”	2	The record name
5	0xD1	1	NDEF header. TNF = 0x01, SR=1, MB=1, ME=1
6	0x01	1	Record name length (1 byte)
7	0x0E	1	The length of the URI payload (14 bytes)
8	“U”	1	Record type: “U”
9	0x01	1	Abbreviation: “http://www.”
10	“nfc-forum.org”	13	The URI itself.

A.2 A URI with a Title and Launching a Browser

In this example, the NDEF message consists of an action record, and it also has two title records to be displayed. It is up to the device to decide which title record to show. For example, if the current locale of the device is Finnish, then a well-designed device would show the Finnish language version of the text; otherwise, it would default to English.

This example displays the use of the non-short message format, even if strictly speaking it is not necessary due to the short lengths of the things involved. You can mix short and long records.

The contents of the NDEF message would be as follows (total length 69 bytes):

Table 5. Example for a Complex URI

Offset	Content	Length	Explanation
0	0xD1	1	NDEF header. TNF = 0x01 (Well Known Type). SR=1, MB=1, ME=1
1	0x02	1	Record name length (2 bytes)
2	0x49	1	Length of the Smart Poster data (73 bytes)
3	“Sp”	2	The record name
5	0x81	1	NDEF header. TNF = 0x01, SR=0, MB=1, ME=0
6	0x01	1	Record name length (1 byte)
7	0x00, 0x00, 0x00, 0x0E	4	The length of the URI payload (14 bytes) (long format)
11	“U”	1	Record type: “U”
12	0x01	1	Abbreviation: “http://www.”
13	“nfc-forum.org”	13	The URI itself.
26	0x11	1	NDEF record header (SR=1, TNF=0x01)
27	0x03	1	The length of the record name
28	0x01	1	The length of the “act” payload.
29	“act”	3	Record type: “act”
32	0x00	1	Action = Launch browser
33	0x11	1	NDEF record header (SR=1, TNF=0x01)
34	0x01	1	Length of the record name
35	0x12	1	Length of the record payload (18 bytes)
36	“T”	1	Record type: “T” (=Text)
37	0x05	1	Status byte for the Text (UTF-8, five-byte code)
38	“en-US”	5	ISO Language code: US-English
43	“Hello, world”	12	The text: “Hello world”, encoded in UTF-8.

Offset	Content	Length	Explanation
55	0x51	1	NDEF record header (SR=1, TNF= 0x01, ME=1)
56	0x01	1	Record name length
57	0x13	1	Length of the Text payload (19 bytes)
58	“T”	1	The name of the Text record (“T”)
59	0x02	1	Status byte: UTF-8, two-byte language code
60	“fi”	2	ISO two-character language code: Finnish
62	“Morjens, maailma”	16	The text “Morjens, maailma” encoded in UTF-8

B. Revision History

The following table outlines the revision history of Smart Poster Record Type Definition.

Table 6. Revision History

Document Name	Revision and Release Date	Status	Change Notice	Supersedes
NFCForum-SmartPoster_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

RESTRICTIONS ON USE

This specification is copyright © 2005-2006 by the NFC Forum, and was made available pursuant to a license agreement entered into between the recipient (Licensee) and NFC Forum, Inc. (Licensor) and may be used only by Licensee, and in compliance with the terms of that license agreement (License). If you are not the Licensee, you are not authorized to make any use of this specification. However, you may obtain a copy at the following page of Licensor's Website: http://www.nfc-forum.org/resources/spec_license after entering into and agreeing to such license terms as Licensor is then requiring. On the date that this specification was downloaded by Licensee, those terms were as follows:

1. LICENSE GRANT.

Licensor hereby grants Licensee the right, without charge, to copy (for internal purposes only) and share the Specification with Licensee's members, employees and consultants (as appropriate). This license grant does not include the right to sublicense, modify or create derivative works based upon the Specification.

2. NO WARRANTIES.

THE SPECIFICATION IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY, COMPLETENESS AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL LICENSOR, ITS MEMBERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE SPECIFICATION.

3. THIRD PARTY RIGHTS.

Without limiting the generality of Section 2 above, LICENSOR ASSUMES NO RESPONSIBILITY TO COMPILE, CONFIRM, UPDATE OR MAKE PUBLIC ANY THIRD PARTY ASSERTIONS OF PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS THAT MIGHT NOW OR IN THE FUTURE BE INFRINGED BY AN IMPLEMENTATION OF THE SPECIFICATION IN ITS CURRENT, OR IN ANY FUTURE FORM. IF ANY SUCH RIGHTS ARE DESCRIBED ON THE SPECIFICATION, LICENSOR TAKES NO POSITION AS TO THE VALIDITY OR INVALIDITY OF SUCH ASSERTIONS, OR THAT ALL SUCH ASSERTIONS THAT HAVE OR MAY BE MADE ARE SO LISTED.

4. TERMINATION OF LICENSE.

In the event of a breach of this Agreement by Licensee or any of its employees or members, Licensor shall give Licensee written notice and an opportunity to cure. If the breach is not cured within thirty (30) days after written notice, or if the breach is of a nature that cannot be cured, then Licensor may immediately or thereafter terminate the licenses granted in this Agreement.

5. MISCELLANEOUS.

All notices required under this Agreement shall be in writing, and shall be deemed effective five days from deposit in the mails. Notices and correspondence to the NFC Forum address as it appears below. This Agreement shall be construed and interpreted under the internal laws of the United States and the Commonwealth of Massachusetts, without giving effect to its principles of conflict of law.

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

Contents

1 Overview	1
1.1 Objectives	1
1.2 Purpose	1
1.2.1 Mission Statement and Goals	1
1.3 References	1
1.4 Administration	1
1.5 Special Word Usage	2
1.6 Name and Logo Usage	2
1.7 Intellectual Property	2
1.8 Acronyms	2
2 Text Record.....	3
2.1 Introduction	3
2.2 Dependencies.....	3
2.3 Security Considerations.....	3
3 NDEF structure	4
3.1 Messaging Sequence	4
3.2 Records Mapping	4
3.2.1 Syntax.....	4
3.2.2 Structure	5
3.3 Language Codes	5
3.4 UTF-16 Byte Order	5
A. Example UTF-8 Encoding	6
B. Revision History	7

Tables

Table 1. Acronyms	2
Table 2. Text Record Content Syntax	4
Table 3. Status Byte Encodings.....	4
Table 4. Example: “Hello, world!”	6
Table 5. Revision History.....	7

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

RESTRICTIONS ON USE

This specification is copyright © 2005-2006 by the NFC Forum, and was made available pursuant to a license agreement entered into between the recipient (Licensee) and NFC Forum, Inc. (Licensor) and may be used only by Licensee, and in compliance with the terms of that license agreement (License). If you are not the Licensee, you are not authorized to make any use of this specification. However, you may obtain a copy at the following page of Licensor's Website: http://www.nfc-forum.org/resources/spec_license after entering into and agreeing to such license terms as Licensor is then requiring. On the date that this specification was downloaded by Licensee, those terms were as follows:

1. LICENSE GRANT.

Licensor hereby grants Licensee the right, without charge, to copy (for internal purposes only) and share the Specification with Licensee's members, employees and consultants (as appropriate). This license grant does not include the right to sublicense, modify or create derivative works based upon the Specification.

2. NO WARRANTIES.

THE SPECIFICATION IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY, COMPLETENESS AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL LICENSOR, ITS MEMBERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE SPECIFICATION.

3. THIRD PARTY RIGHTS.

Without limiting the generality of Section 2 above, LICENSOR ASSUMES NO RESPONSIBILITY TO COMPILE, CONFIRM, UPDATE OR MAKE PUBLIC ANY THIRD PARTY ASSERTIONS OF PATENT OR OTHER INTELLECTUAL PROPERTY RIGHTS THAT MIGHT NOW OR IN THE FUTURE BE INFRINGED BY AN IMPLEMENTATION OF THE SPECIFICATION IN ITS CURRENT, OR IN ANY FUTURE FORM. IF ANY SUCH RIGHTS ARE DESCRIBED ON THE SPECIFICATION, LICENSOR TAKES NO POSITION AS TO THE VALIDITY OR INVALIDITY OF SUCH ASSERTIONS, OR THAT ALL SUCH ASSERTIONS THAT HAVE OR MAY BE MADE ARE SO LISTED.

4. TERMINATION OF LICENSE.

In the event of a breach of this Agreement by Licensee or any of its employees or members, Licensor shall give Licensee written notice and an opportunity to cure. If the breach is not cured within thirty (30) days after written notice, or if the breach is of a nature that cannot be cured, then Licensor may immediately or thereafter terminate the licenses granted in this Agreement.

5. MISCELLANEOUS.

All notices required under this Agreement shall be in writing, and shall be deemed effective five days from deposit in the mails. Notices and correspondence to the NFC Forum address as it appears below. This Agreement shall be construed and interpreted under the internal laws of the United States and the Commonwealth of Massachusetts, without giving effect to its principles of conflict of law.

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

Contents

1 Overview	1
1.1 Objectives	1
1.2 Purpose	1
1.2.1 Mission Statement and Goals	1
1.3 References	1
1.4 Administration	1
1.5 Special Word Usage	2
1.6 Name and Logo Usage	2
1.7 Intellectual Property	2
1.8 Acronyms	3
2 URI Service.....	4
2.1 NDEF Message Sequences	4
2.2 Dependencies.....	4
3 NDEF Structure.....	5
3.1 Messaging Sequence	5
3.2 Records Mapping	5
3.2.1 URI Record Type	5
3.2.2 URI Identifier Code.....	5
3.2.3 URI Field.....	7
4 Handling Guideline.....	8
A. Examples.....	9
A.1 Simple URL with No Substitution	9
A.2 Storing a Telephone Number	9
A.3 Storing a Proprietary URI on the Tag.....	10
B. Revision History	11

Tables

Table 1. Acronyms	3
Table 2. URI Record Contents	5
Table 3. Abbreviation Table.....	5
Table 4. Simple URL with No Substitution	9
Table 5. Storing a Telephone Number	9
Table 6. Storing a Proprietary URI on the Tag.....	10
Table 7. Revision History	11

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	btsp://
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	