Signature Record Type Definition

Technical Specification

Version 2.0

2020-02-20

[RTD_SIG]

NFC Forum^{TM}

Contents

Figures

Tables

1 Introduction

Digital signing of NDEF data is a trustworthy method for providing information about the origin of NDEF data in an NFC Forum Tag and NFC Forum Device. Digital signing provides users with the possibility of verifying the authenticity and integrity of the data in an NDEF message.

1.1 Objectives

The Signature Record Type Definition (RTD) specifies the format used when signing single or multiple NDEF records. This specification defines the required and optional signature RTD fields, and provides a list of suitable signature algorithms and certificate types that can be used to create the signature.

This specification does not define or mandate a specific PKI or certification system, nor a new algorithm for use with the Signature RTD. Specification of the certificate verification and revocation process is also out of scope.

1.2 Applicable Documents or References

[DSS]	Digital Signature Standard (DSS), FIPS PUB 186-3, June 2009 Information Technology Laboratory, National Institute of Standards and Technology
[NDEF]	NFC Data Exchange Format Technical Specification, NFC Forum
[IEEE1363]	IEEE Standard Specifications for Public-Key Cryptography, August 2000, The Institute of Electrical and Electronics Engineers, Inc.
[SP800-131A]	Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths, NIST SP 800-131A, January 2011, Information Technology Laboratory, National Institute of Standards and Technology
[PKCS_1]	PKCS#1, RSA Cryptography Standard, Version 2.1, June 2002, RSA Laboratories
[RFC2119]	Key words for use in RFCs to Indicate Requirement Levels, RFC 2119, S. Bradner, March 1997, Internet Engineering Task Force
[RFC8141]	Uniform Resource Names (URNs), RFC 8141, P. Saint-Andre Filament, J. Klensin, April 2017, Internet Engineering Task Force

[RFC3280]	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 3280, April 2002, Internet Engineering Task Force
[RFC3492]	Punycode: A Bootstring Encoding of Unicode for Internationalized Domain Names in Applications (IDNA), RFC 3492, A. Costello, March 2003, Internet Engineering Task Force
[RFC3987]	Internationalized Resource Identifiers (IRIs), RFC 3987, M. Duerst, M. Suignard, January 2005 Microsoft Corporation
[RTD]	Record Type Definition Technical Specification, NFC Forum
[SEC1]	Standards for Efficient Cryptography 4 (SEC1): Elliptic Curve Cryptography, Version 1.0, September 20, 2000, Certicom Research
[SEC4]	Standards for Efficient Cryptography 4 (SEC4): Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV), Version 1.0, January 2013, Certicom Research
[SHS]	Secure Hash Standard (SHS), FIPS PUB 180-3, October 2008, Information Technology Laboratory, National Institute of Standards and Technology
[URI_RTD]	URI Record Type Definition Technical Specification, NFC Forum
[X_509]	Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, ITU-T Recommendation, August 2005, International Telecommunication Union (ITU) Telecommunication Standardization Sector (ITU-T)

1.3 Administration

The NFC Forum Signature Record Type Definition 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/

This specification is maintained by the NFC Forum, Inc.

1.4 Trademark and Logo Usage

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

1.5 Intellectual Property

The Signature Record Type Definition Technical Specification conforms to the Intellectual Property guidelines specified in the NFC Forum *Intellectual Property Rights Policy*, as outlined in the NFC Forum *Rules of Procedure*. These documents are available on the NFC Forum website.

1.6 Special Word Usage

The key words "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT" and "MAY" in this document are to be interpreted as described in [RFC2119].

1.7 Abbreviations

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

Table 1: Abbreviations

Abbreviation	Description
CA	Certificate Authority
DSA	Digital Signature Algorithm
ECDSA	Elliptic Curve Digital Signature Algorithm
NDEF	NFC Data Exchange Format
PKI	Public Key Infrastructure
RFU	Reserved for Future Use
RSA	Rivest-Shamir-Adleman encryption algorithm (public key encryption algorithm)
RTD	Record Type Definition
SHA-256	Secure Hash Algorithm
URI	Uniform Resource Identifier (e.g., http://, ftp://, mailto:, news:)
URL	Uniform Resource Locator (a special case of a URI)
URN	Uniform Resource Name. A particular type of URI that is defined in [RFC8141].

1.8 Glossary

Application User

A person who interacts with an application based on this specification through a user interface.

Big Endian

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

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.

NFC Forum Device

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

NFC Reader Device

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

NFC Tag Device

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

NFC Universal Device

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

2 Signature Record Overview

2.1 Introduction

The Signature record contains a digital signature related to one or more NDEF records within an NDEF message. The signature can be used to verify the integrity and authenticity of the content, i.e., the NDEF records that have been signed.

2.2 Dependencies

There are no specific dependencies for the Signature RTD.

2.3 Security Considerations

The primary function of the Signature record is to verify the integrity and authenticity of data (i.e., certain NDEF record(s) or the whole NDEF message) in an NFC Forum Device.

However, a malicious third party could delete the Signature record from the NDEF message or attach a new Signature record to prevent the application user from noticing any malicious change of content. It must be understood that the verification is only as trustworthy as the tools (signature algorithm, certificate, etc.) and processes (e.g., security policies) that are being used. These risks, along with the use of the Signature record, should be taken into consideration in the development of applications.

3 Signature NDEF Structure

3.1 Messaging Sequence

There is no particular messaging sequence.

3.2 Use of RFU Fields and Values

This document defines some of the Signature, Hash, and Certificate type values as Reserved for Future Use (RFU). The basic rules for RFU values are:

- An implementation of this version of the Signature RTD Specification SHALL NOT send a Signature, Hash, or Certificate type value defined as RFU.
- An implementation of this version of the Signature RTD Specification that receives a Signature, Hash or Certificate type value defined as RFU SHALL ignore the Signature record in which the value was contained and SHALL consider the signature as invalid.

3.3 Records Mapping

3.3.1 Syntax

The NFC Forum Well Known Type ([NDEF], [RTD]) for the Signature record is "Sig" (0x53, 0x69, 0x67).

The contents of the payload of a Signature record consist of the following fields: Version, Signature and Certificate Chain. The record is illustrated in Table 2.

Table 2: Signature Record

Signature Record		
Version	Signature	Certificate Chain

3.3.2 Version Field

The Version field, as shown in Table 3, is a 1-octet field. The value of this field SHALL be set equal to the version number of the Signature RTD specification, after which the Signature Record is encoded. The version number is divided into a major part and a minor part. The higher 4 bits of the Version field SHALL encode the major version as a 4-bit integer. The lower 4 bits of the Version field SHALL encode the minor version as a 4-bit integer.

Table 3: Signature Record Version Field

7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Major Version				Minor Version			

Signatures created to be compliant with this specification SHALL have their Version fields set to 2.0 (0x20).

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. Parsers that support a higher major version MAY also provide support for previous versions. Parsers that do not implement the major version number set in the Version field SHALL NOT further interpret the data.

During verification of Signature records, the following rules regarding backward-compatibility between minor versions SHALL apply:

- If the major version number is supported by the parser and the minor version number is equal to or lower than that supported by the parser, the Signature record SHALL be treated as fully compatible to the version implemented by the parser.
- If the major version number is supported by the parser and the minor version number is higher than that supported by the parser, the Signature record SHALL be treated as compatible to the version implemented by the parser. Signature records containing RFU fields or values SHALL be ignored. RFU treatment according to Section 3.2 SHALL NOT apply to these records.
- The Version field value 0x01 is obsolete. Devices implementing this specification to create Signature records SHALL NOT use an obsolete version value. Devices implementing this specification to verify signatures SHALL ignore all Signature records that have an obsolete version value.

3.3.3 Signature Field

The Signature field contains either the actual signature or a reference to the location where the signature can be found. The Signature field contains up to five subfields and SHALL be formatted as shown in Table 4.

Table 4: Signature Field Subfields

Signature Field							
URI_Present	**Signature Type**	**Hash Type**	**Signature / URI Length**		**Signature / URI**		
1 bit	7 bits	8 bits	16 bits		N octets		
Octet 0		Octet 1	Octet 2	Octet 3	Octet 4	…	Octet (3+N)

The URI_Present flag SHALL be a 1-bit field indicating whether a signature or a reference to the signature is present in the record.

- If URI_Present = 0 and Signature Type = 0, then the Signature record SHALL NOT be used to verify the preceding NDEF record(s) from the beginning of the NDEF message or the previous Signature record. In this case, the Hash Type subfield, the Signature / URI Length subfield and the Signature / URI subfields SHALL NOT be present and the Certificate Chain field SHALL NOT be present. In this case, the Signature record is used as a start marker to indicate the beginning of a collection of NDEF records to which a subsequent Signature record will be applied. See Section 3.4.
- If URI_Present is set to 0 and the Signature Type is not set to 0, then the Hash Type subfield SHALL contain a hash method according to Table 6, the Signature / URI subfield SHALL contain the actual signature and the Signature / URI Length subfield SHALL contain a 16-bit unsigned number denoting the length, in octets, of that signature.
- If URI_Present is set to 1 and the Signature Type is not set to 0, then the Hash Type subfield SHALL contain a hash method according to Table 6, the Signature / URI subfield SHALL contain a URI that is a reference to the signature location and the Signature / URI Length subfield SHALL contain a 16-bit unsigned number denoting the length, in octets, of that URI. The URI SHALL be in the format specified in Section 3.3.3.4.
- If URI_Present is set to 1, the Signature Type SHALL NOT be set to 0.

3.3.3.1 Signature Type / Hash Type

The Signature Type SHALL be a 7-bit field indicating the type of the signature algorithm and SHALL use values defined in Table 5. Each signature type value indicates an explicit key length.

Table 5: Signature Type Values

Hex	Signature Type & Key Length	Security Strength[1]
0x00	No signature present	Not applicable
0x01	RSASSA-PSS [PKCS_1] 1024	80 bits
0x02	RSASSA-PKCS1-v1_5 [PKCS_1] 1024	
0x03	DSA [DSS] 1024	
0x04	ECDSA [DSS] P192	
0x05	RSASSA-PSS [PKCS_1] 2048	112 bits
0x06	RSASSA-PKCS1-v1 [PKCS_1] 2048	
0x07	DSA [DSS] 2048	
0x08	ECDSA [DSS] P224	
0x09	ECDSA [DSS] K233	
0x0a	ECDSA [DSS] B233	
0x0b	ECDSA [DSS] P256	128 bits
0x0c – 0x7f	RFU	Not applicable

[1] Security strength based on the algorithm and the key length, according to [IEEE1363].

The Hash Type SHALL be an 8-bit field indicating the type of the hash algorithm and SHALL use values defined in Table 6.

Table 6: Hash Type Values

Hex	Hash Type
0x00	RFU
0x01	RFU
0x02	SHA-256 [SHS]
0x03 – 0xff	RFU

3.3.3.2 Signature / URI Length

The Signature / URI Length field SHALL be a 16-bit field that indicates the number of octets in the Signature /URI field. The value of the Signature / URI Length field SHALL be coded in Big Endian format.

3.3.3.3 Signature Format

When URI_Present is set to 0, the Signature / URI subfield provides the Signature. The signature shall be a single octet string when the Signature Type field refers to an RSASSA signature and a concatenation of two octet strings, ‘r’ followed by ‘s’ as per [DSS], when the Signature Type field refers to a DSA or an ECDSA signature. In the second case, the two octet strings are of the same length, which is equal to the byte length of the order of the field generator, and is half the length indicated in the Signature / URI Length field. When RSASSA, DSA and ECDSA signatures are generated, the integer values are converted to octet strings with the I2OSP primitive, as per [IEEE1363]. When RSASSA, DSA and ECDSA signatures are being validated, the octet strings are converted to integers with the OS2IP primitive, as per [IEEE1363].

3.3.3.4 URI Format

When URI_Present is set to 1, the Signature / URI subfield provides the URI, as per [RFC3987] (so it is actually an Internationalized Resource Identifier (IRI), but for legacy reasons we use the term URI). This URI can be a URL or URN. The encoding used SHALL be UTF-8, unless the URI scheme specifies a particular encoding. Most modern applications support URI.

The length is in octets, not in characters, because 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 [RFC3987] and IDN [RFC3492]. For different schemes, the encoding might be different.

For example, if the URI contains the string “http://www.hääyö.com/”, it is transformed, as per standard IDN [RFC3492] rules, into “http://www.xn--hy-viaa5g.com” before acting on it. The implementations SHOULD include support for IRI where display of the URI in human-readable form is anticipated.

In the URI any character value from 0 through 31 SHALL be recorded as an error, and the URI record SHALL be discarded. Any invalid UTF-8 sequence SHALL be considered an error, and the entire URI record SHALL be discarded.

The URI SHALL refer to a signature formatted in the same way as the Signature Format in Section 3.3.3.2, which can be further encoded in a text encoding, e.g. base 64.

3.3.4 Certificate Chain Field

The Certificate Chain field contains mandatory and optional subfields, as defined in Table 7.

Table 7: Certificate Chain Field Subfields

Certificate Chain Field						
URI_Present	**Cert_Format**	**Nbr_of_Certs**	**Cert_Store**			**Cert_URI**
1 bit	3 bits	4 bits	Certificate_0	…	Certificate_n	URI Field
Octet 0			Octet 1 …			

The URI_Present flag SHALL be a 1-bit field indicating whether a certificate or a reference to the certificate is present in the record.

- If URI_Present is set to 1, the Cert_URI subfield SHALL be present.
- If URI_Present is set to 0, the Cert_URI subfield SHALL NOT be present.

The Cert_Format field SHALL be a 3-bit field that indicates the type of certificate present in the Signature record and SHALL use one of the values defined in Table 8.

Table 8: Certificate Format Values

Hex	**Certificate Format**
0x00	X.509 [X_509][Annex B]
0x01	M2M [Annex A]
0x02-0x07	RFU

The Nbr_of_Certs SHALL be a 4-bit integer that specifies only the number of certificate subfields that are present in the Cert_Store field and SHALL have a value in the range of 0 to 15.

- If the Nbr_of_Certs is greater than 0, then the first subfield in Cert_Store SHALL be the signer's certificate and SHALL then be followed by zero or more certificates. Each following certificate SHALL directly certify the one preceding it.

The Cert_Store subfield SHALL NOT contain the top-most certificate in the certificate hierarchy (e.g., the root Certificate Authority certificate in an X.509 certificate chain).

The top-most certificate in the certificate hierarchy SHALL NOT be present in the chain referenced by the Cert_URI subfield.

If present, the Cert_URI SHALL contain a URI that is a reference to the next certificate in the chain following the last certificate contained in the Cert_Store.

If a signature is given within the Signature field, either by inclusion or by reference, and the URI_Present and Nbr_of_Certs are set to 0 in the Certificate Chain field, then the identification and retrieval of the certificate chain is out of scope of this document.

Because certificate validation requires that the top-most certificate in the certificate hierarchy be distributed independently, this specification omits the top-most certificate from both the Cert_Store and Cert_URI subfields, under the assumption that the NFC Forum Device already possesses it in order to validate the chai

3.3.4.1 Certificate Subfield

The format for the Certificate subfield in the Cert_Store field SHALL be as shown in Table 9.

Table 9: Certificate Format

Certificate	
Length, N	**Value**
16 bits	N octets

- The Length field SHALL be a 16-bit field that indicates the number of octets in the Value field. The value of the Length field SHALL be coded in Big Endian format.
- The Value field SHALL contain the certificate.

3.3.4.2 URI Subfield

The format for the URI subfield SHALL be as shown in Table 10.

Table 10: URI Subfield Format

URI	
Length, N	**Value**
16 bits	N octets

- The Length field SHALL be a 16-bit field that indicates the number of octets in the Value field. The value of the Length field SHALL be coded in Big Endian format.
- The Value field SHALL contain a URI as defined in Section 3.3.3.4.
- The URI SHALL refer to one or more certificates formatted in the same way as the Certificate Subfield in Section 3.3.4.1, i.e. 16-bit field indicating the length of the first cert then the first cert, and similarly for all the remaining certs. This data can be further encoded in a text encoding, e.g. base 64.

3.4 Use of the Signature Record within an NDEF Message

The Signature Record SHALL apply to all preceding records, starting either from the first NDEF record of the NDEF message or from the first NDEF record following the preceding Signature Record. The Signature Record itself is not signed.

The Signature Record SHALL apply to the entire NDEF record, including the first byte of the NDEF header. Note that the MB and ME flags will have to be adjusted appropriately to accommodate the addition of the Signature record (see Figure 1).

When an application needs to sign only a selection of NDEF records within the NDEF message, an empty Signature record MAY be inserted in the records to act as a start marker as illustrated in Figure 1. Also, see Section 3.3.3.

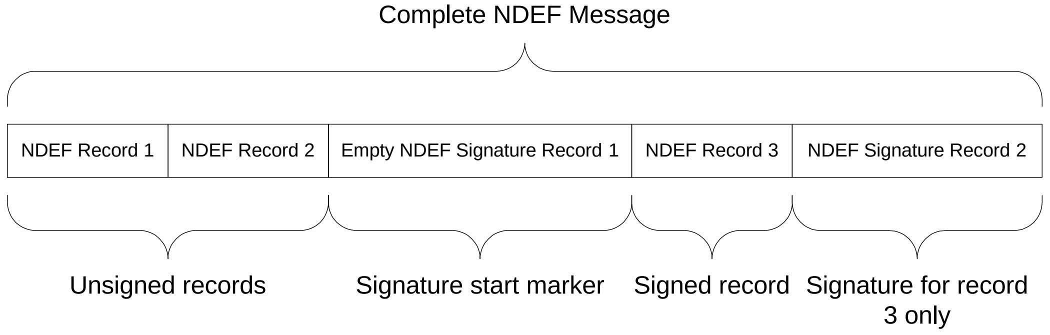

Figure 1: Example of the Use of an Empty Signature record

When a signature is applied to a NDEF record or selection of NDEF records, then the rules for concatenating NDEF records SHALL be followed, which might involve changing the state of MB, ME, SR, and length fields. These changes SHALL be completed before computing the signature.

If any NDEF records included in a signature are changed, or if the order of NDEF records is changed, then the signature will be affected.

3.5 Use of Signature Record with Data Other than an NDEF Message

While this specification does not preclude the use of the Signature record outside the NDEF framework, its use in such applications is not specified in this document.

A. Exhibit A

No items have been included in Exhibit A.

B. M2M Certificate Profile

B.1 Machine-to-Machine Certificate Format

All Certificates should use the Machine-to-Machine certificate format below, as specified in ASN.1 notation.

```
M2M-Certificate-Definition
    {joint-iso-ccitt (2) country (16) usa (840) us-company (1)
     nfc-forum (114513) modules (5) m2m-certificate (0)}

-- Structure Must be DER encoded
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

-- The APPLICATION 20 tag is intended to make the M2M format apparent
-- by inspecting the first octet of the encoding

Certificate ::= [APPLICATION 20] IMPLICIT SEQUENCE {
    tbsCertificate     TBSCertificate, -- To be signed certificate
    cACalcValue        OCTET STRING -- Contains signature for a signed
                       -- certificate or public key derivation value
                       -- for an ECQV cert, see B.2
}

TBSCertificate ::= SEQUENCE {
    version            INTEGER {v1(0)} DEFAULT v1,
    serialNumber       OCTET STRING (SIZE (1..20)),
    cAAlgorithm        OBJECT IDENTIFIER OPTIONAL, -- Identifies
                       -- CA algorithm, hash function, and
                       -- optionally other required parameters
                       -- (e.g. in the case of ECC, the curve).
                       --
                       -- Required for signature verification but may
                       -- be omitted from the transmitted cert and
                       -- filled in from the pKAlgorithm of a
                       -- superior cert (provided not root cert) prior
                       -- to signature verification.
                       -- For omitting rules, see B.3
                       --
    cAAlgParams        OCTET STRING OPTIONAL, -- Identifies
                       -- CA algorithm parameters
                       --
                       -- This specification does not provide for
                       -- omitting this field in transmission and
                       -- subsequently replacing it from a superior
                       -- certificate for signature verification
                       --
    issuer             Name OPTIONAL, -- Identifies
                       -- issuer name
                       --
                       -- Required for signature verification but may
                       -- be omitted from the transmitted cert and
                       -- filled in from the subject field of a
                       -- superior cert (provided not root cert) prior
                       -- to signature verification
                       --
    validFrom          OCTET STRING (SIZE (4..5)) OPTIONAL,
                       -- Unix time. If omitted no validity specified
    validDuration      OCTET STRING (SIZE (1..4)) OPTIONAL,
                       -- # of seconds. If omitted no expiry specified
    subject            Name,
    pKAlgorithm        OBJECT IDENTIFIER OPTIONAL, -- Default is
                       -- same as cAAlgorithm in this certificate.
                       -- For omitting rules, see B.3
pKAlgParams        OCTET STRING OPTIONAL,
    pubKey             OCTET STRING OPTIONAL,
                       -- Omit for an ECQV certificate, see A.2
    authKeyId          AuthKeyId OPTIONAL,
    subjKeyId          OCTET STRING OPTIONAL,
    keyUsage           OCTET STRING (SIZE (1)) OPTIONAL, -- Critical
                       -- One byte containing a bit string,
                       -- as described below
    basicConstraints   INTEGER (0..7) OPTIONAL, -- If absent this is
                       -- an end-entity cert; max intermed path length
                       -- for CA certificate
    certificatePolicy  OBJECT IDENTIFIER OPTIONAL, -- May use the
                       -- current version of this policy
    subjectAltName     GeneralName OPTIONAL,
    issuerAltName      GeneralName OPTIONAL,
    extendedKeyUsage   OBJECT IDENTIFIER OPTIONAL,
    authInfoAccessOCSP IA5String OPTIONAL,-- OCSP responder URI
    cRLDistribPointURI IA5String OPTIONAL,-- CRL distribution point URI
    x509extensions     X509Extensions OPTIONAL,
    ...
}
Name ::= SEQUENCE SIZE (1..4) OF AttributeValue

AttributeValue ::= CHOICE {
    country            PrintableString (SIZE (2)),
    organization       UTF8String (SIZE (1..32)),
    organizationalUnit UTF8String (SIZE (1..32)),
    distinguishedNameQualifier PrintableString (SIZE (1..32)),
    stateOrProvince    UTF8String (SIZE (1..4)),
    locality           UTF8String (SIZE (1..32)),
    commonName         UTF8String (SIZE (1..32)),
    serialNumber       PrintableString (SIZE (1..32)),
    domainComponent    IA5String (SIZE (1..32)),
    registeredId       OBJECT IDENTIFIER,
    octetsName         OCTET STRING (SIZE (1..8))
}
AuthKeyId ::= SEQUENCE {
    keyIdentifier      OCTET STRING OPTIONAL,
    authCertIssuer     GeneralName OPTIONAL,
    authCertSerialNum  OCTET STRING (SIZE(1..20)) OPTIONAL
}
X509Extensions ::= SEQUENCE OF Extension

Extension ::= SEQUENCE {
        extnID         OBJECT IDENTIFIER,
        criticality    BOOLEAN DEFAULT FALSE,
        extnValue      OCTET STRING
```

```
}
GeneralName ::= CHOICE {
    rfc822Name         IA5String (SIZE (1..128)),
    dNSName            IA5String (SIZE (1..128)),
    directoryName      Name,
    uniformResourceIdentifier IA5String (SIZE (1..128)),
    iPAddress          OCTET STRING (SIZE (1..16)),
                       --4 octets for IPV4 16 octets for IPV6
    registeredID       OBJECT IDENTIFIER
}

-- Notes:
-- * The times are represented using UNIX time, i.e. # of seconds
--   since the unix epoch: http://en.wikipedia.org/wiki/Unix_time
--   The validFrom field permits 40-bit values to avoid problems in
--   2038 (when 32-bit values won't be enough).
--
-- * The keyUsage field conveys a single octet equal to the
--   second octet of the DER encoding of the following BIT STRING
--
--   KeyUsage ::= BIT STRING {
--       digitalSignature (0),
--       nonRepudiation (1),
--       keyEncipherment (2),
--       dataEncipherment (3),
--       keyAgreement (4),
--       keyCertSign (5),
--       Use keyCertSign also for an ECQV certificate issuer
--       cRLSign (6)
--       the last bit in the byte is always zero (7)
--   }
--

END
```

A CA shall use a M2M certificate profile that consists of the following information:

- ***serialNumber***: This field must be present and must contain a unique serial number containing at least 20 randomly generated bits.
- ***issuer***: The issuer organization name and country fields must be present and must contain information that accurately identifies the issuing CA. All other Issuer fields should not be present.
- ***subject***: The *organization*, *stateOrProvince* or *locality*, and *country* fields must be present. The *locality* is only required if the *stateOrProvince* is not present, and the *stateOrProvince* is only required of the locality is not present. All fields must contain information verified in accordance with the NFC Forum Signature RTD Certificate Policy. The organization field may contain the name of an individual.

  If present, the *commonName* field must contain either the Applicant's verified name or an identifier for the subject device.

  All other subject fields, including *OU* fields, should not be present. If present, the field must contain information verified by the CA or RA in accordance with this the NFC Forum Signature RTD Certificate Policy.

- ***subjectAltName***: This field must not be present.
- ***authInfoAccessOCSP***: This field should not be present. If present, it must contain the HTTP URL of the issuer's OCSP responder.
- ***cRLDistribPointURI***: This field must be present and must contain the HTTP URL of the CRL that will list the certificate if the certificate is ever revoked.
- ***keyUsage***: The bits for *digitalSignature* and *keyEncipherment* must be set. The bit for *keyAgreement* may be set. All other bits should not be set.
- ***extendedKeyUsage***: The extended key usage extension must include the NFC Forum's OID [2.16.840.1.114513.29.37]. This OID serves as a policy identifier that asserts the certificate as compliant with the NFC Forum Signature RTD Certificate Policy and as an extended key usage for creating Signatures.

B.2 *cACalcValue* and *pubKey* Fields in Machine-to-Machine Certificate Format

Following is the definition of how the fields *cACalcValue* and *pubKey* are filled in an M2M certificate for the set of algorithms identified in the NFC Forum Signature RTD Certificate Policy. Both fields are of ASN.1 type OCTET STRING (note that analogous strings in X.509 are BIT STRING).

Contents of *cACalcValue* Field:

- ECDSA Algorithms

  This field conveys a digital signature as a DER encoding of a value of the following ASN.1 type (compatible with [SEC1]):

```
ECDSA-Signature ::= CHOICE {
    two-ints-plus ECDSA-Sig-Value,
    point-int [0] ECDSA-Full-R,
    ... -- Future representations may be added
}
ECDSA-Sig-Value ::= SEQUENCE {
    r INTEGER,
    s INTEGER,
    a INTEGER OPTIONAL,
    y CHOICE { b BOOLEAN, f FieldElement } OPTIONAL – See [SEC1]
}
ECDSA-Full-R ::= SEQUENCE {
    r ECPoint, -- Elliptic curve point represented as an
               -- OCTET STRING determined following the procedure
               -- defined in Clause 2.3.3 of [SEC1]
    s INTEGER
}
```

- ECQV Algorithms

  This field conveys an ECQV public key reconstruction value specifying an elliptic curve point represented as an OCTET STRING determined following the procedure defined in Clause 2.3.3 of [SEC1].

- RSA Algorithms

  This field conveys a RSA digital signature as the DER encoding OCTET STRING value.

  ```
  RSA-Signature ::= OCTET STRING
  ```

Contents of *pubKey* Field:

- ECDSA Algorithms

  This field conveys a public key comprising an elliptic curve point represented as an OCTET STRING determined following the procedure define in [SEC1] Clause 2.3.3.

- ECQV Algorithms

  For ECQV algorithms, this field is omitted.

- RSA Algorithms

  This field contains an RSA public key as a DER encoding of a value of the following ASN.1 type (see [RFC3447], Section A.1.1).

  ```
  RSAPublicKey ::= SEQUENCE {
      modulus           INTEGER, -- n
      publicExponent    INTEGER  -- e
  }
  ```

B.3 Rules for Omitting Algorithm Fields in Machine-to-Machine Certificates

Following are the rules defining when and how omitting algorithm fields is allowed. It is important to differentiate CA certificates and end entity certificates with regards to omitting the algorithm fields.

Algorithm fields in CA certificates:

- ***cAAlgorithm***: Omitting is only allowed when *pKAlgorithm* (or *cAAlgorithm* if *pKAlgorithm* is omitted in a superior certificate) of a superior certificate fully specifies the signature algorithm and its parameters (i.e. signature and hash algorithms plus any required parameters, e.g. in the case of ECC, the curve)
- ***pKAlgorithm***: If omitted in a CA certificate, *cAAlgorithm* specifies the signature & hash algorithms and any required parameters (e.g. curve) for *pubKey*

Algorithm fields in End Entity Certificates:

- ***cAAlgorithm***: Omitting is only allowed when *pKAlgorithm* (or *cAAlgorithm* if *pKAlgorithm* is omitted in a superior certificate) of a superior certificate fully specifies the signature algorithm and its parameters (i.e. signature and hash algorithms plus any required parameters, e.g. in the case of ECC, the curve)
- ***pKAlgorithm***: If omitted in an end entity certificate, *cAAlgorithm* specifies the required parameters (e.g. curve and optionally signature and hash algorithms) for *pubKey*

C. X.509 Certificate Profile

A CA may use a modified X.509 certificate profile that consists of the following information:

- **Serial Number**: This field must be present and must contain a unique serial number containing at least 20 randomly generated bits.
- **Issuer**: The issuer organization name and country fields must be present and must contain information that accurately identifies the issuing CA. All other Issuer fields should not be present.
- **Subject**: The subject:organizationName (OID 2.5.4.10), subject:stateOrProvinceName (OID 2.5.4.8) or subject:localityName (OID 2.5.4.7), and subject:countryName (OID 2.5.4.6) fields must be present. The subject:localityName is only required if the subject:stateOrProvinceName is not present, and the subject:stateOrProvinceName is only required of the subject:localityName is not present. All fields must contain information verified in accordance with the NFC Forum Signature RTD Certificate Policy. The organizationName may contain the name of an individual.

  If present, the subjectCommonName (OID: 2.5.4.6) must contain either the Applicant's verified name or an identifier for the subject device.

  All other subject fields, including OU fields, should not be present. If present, the field must contain information verified by the CA or RA in accordance with this CP.
- **Subject Alternative Name**: This extension must not be present.
- **Authority Information Access**: This extension should not be present. If present, it must contain the HTTP URL of the issuer's OCSP responder.
- **CRL Distribution Points**: This extension must be present and must contain a single distribution point giving the HTTP URL of the CRL that will list the certificate if the certificate is ever revoked.
- **Key Usage**: The bits for digital signature and key encipherment must be set. All other bits should not be set.
- **Extended Key Usage**: The extended key usage extension must include the NFC Forum's OID [2.16.840.1.114513.29.37]. This OID serves as a policy identifier that asserts the certificate as compliant with these guidelines and as an extended key usage for creating Signatures.

D. Recommended Practices

D.1 Usage of URI References

Note that obtaining signatures and certificates from a URI reference poses a potential security risk. The following practice is recommended for implementations:

- Access to certificates from a URI should be approved by the application user.
- URI locations for certificates could be restricted to a list that was obtained by some out-of-band method in a similar fashion as obtaining the root certificate.

D.2 Mixing Signed and Unsigned NDEF Messages

Unsigned NDEF records can easily be manipulated. Therefore, it is recommended that tags should not contain a mix of signed and unsigned NDEF records.

D.3 Multiple Signatures within one Context

Signed NDEF records can be copied and concatenated with other signed records as a possible threat. Therefore, it is recommended that records that form one logical context should be signed with the same Signature record. With records that contain nested NDEF messages, signature of the outer record is preferred over signature of individual records within the nested NDEF message.

E. Examples

E.1 Recommended Usage

E.1.1 Example 1

Figure 2 shows the format of an NDEF message in which there is a Text record ("Hello world") and a URI to a website before a signature is applied.

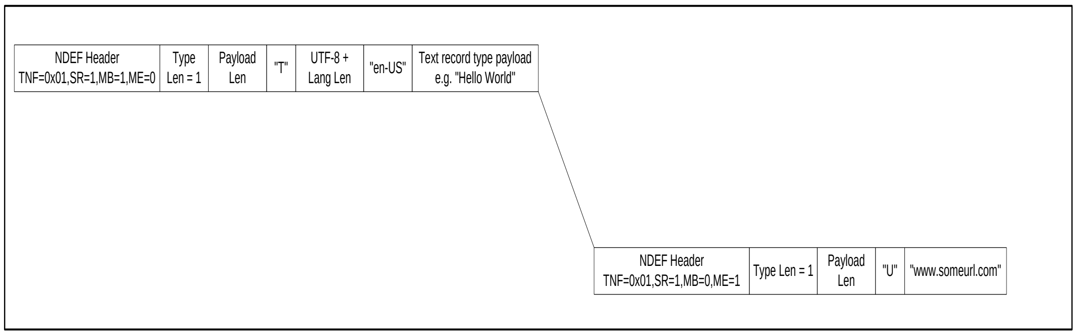

Figure 2: Unsigned Record Example

Figure 3 shows the same NDEF message, but with a signature following the URI record that signs both the Text record and the URI record. Note that before the signature is computed, the ME flag of the URI record must be changed to 0.

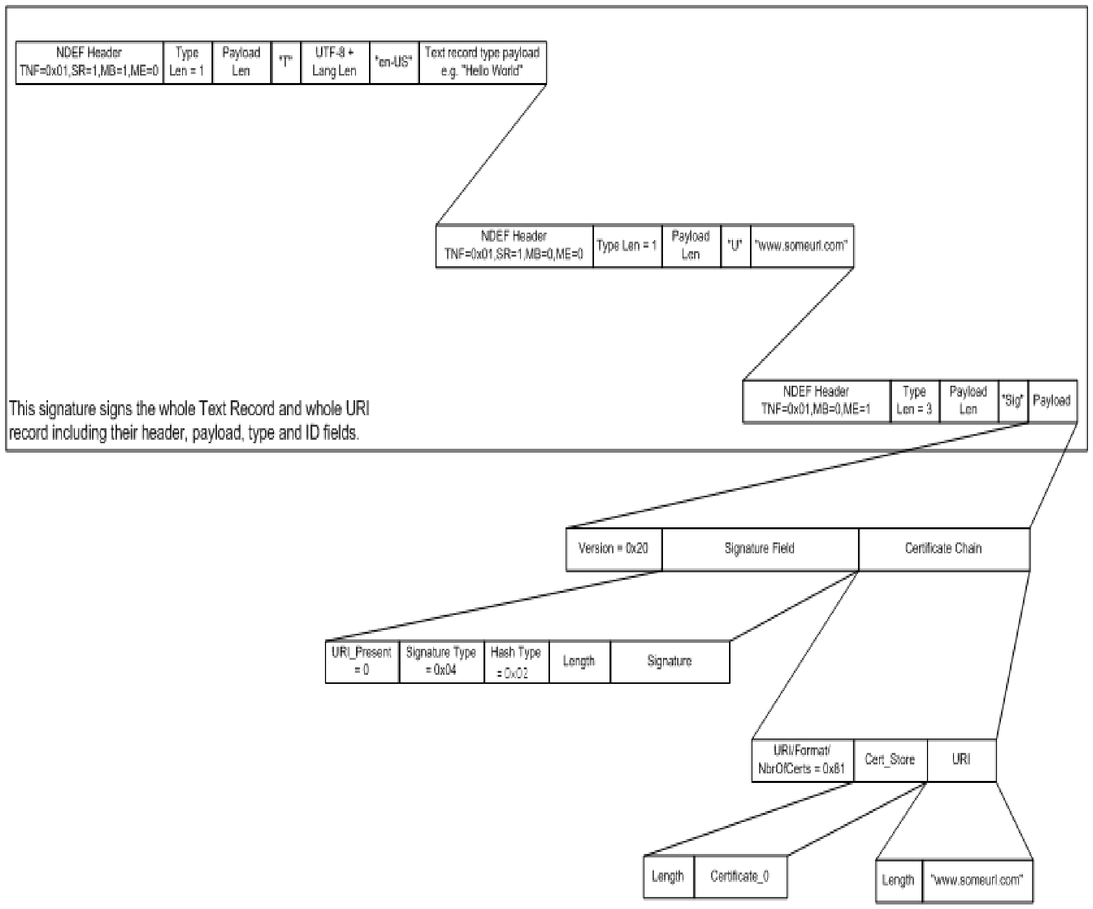

Figure 3: Signed Record Example

The Signature record contains a type 4 signature (ECDSA - P-192), a type 2 hash (SHA-256) and a type 0 (X.509) certificate chain; the certificate chain includes a URI.

E.1.2 Example 2

Figure 4 shows the format of an NDEF message in which there are two MIME records.

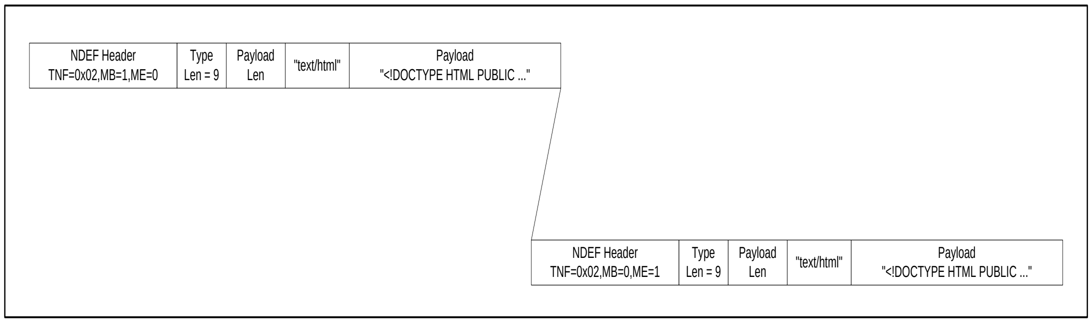
NDEF Header TNF=0x02,MB=1,ME=0
Type Len = 9
Payload Len
"text/html"
Payload "<!DOCTYPE HTML PUBLIC ..."
NDEF Header TNF=0x02,MB=0,ME=1
Type Len = 9
Payload Len
"text/html"
Payload "<!DOCTYPE HTML PUBLIC ..."

Figure 4: Unsigned MIME Record Example

Figure 5 shows the same NDEF message, but with two signatures. The signature following the first MIME record signs only the first MIME record. The signature following the second MIME record signs only the second MIME record. Note that the two MIME records should not share one context, as signing two records that share one context with different Signature records is not recommended (see Appendix D.3).

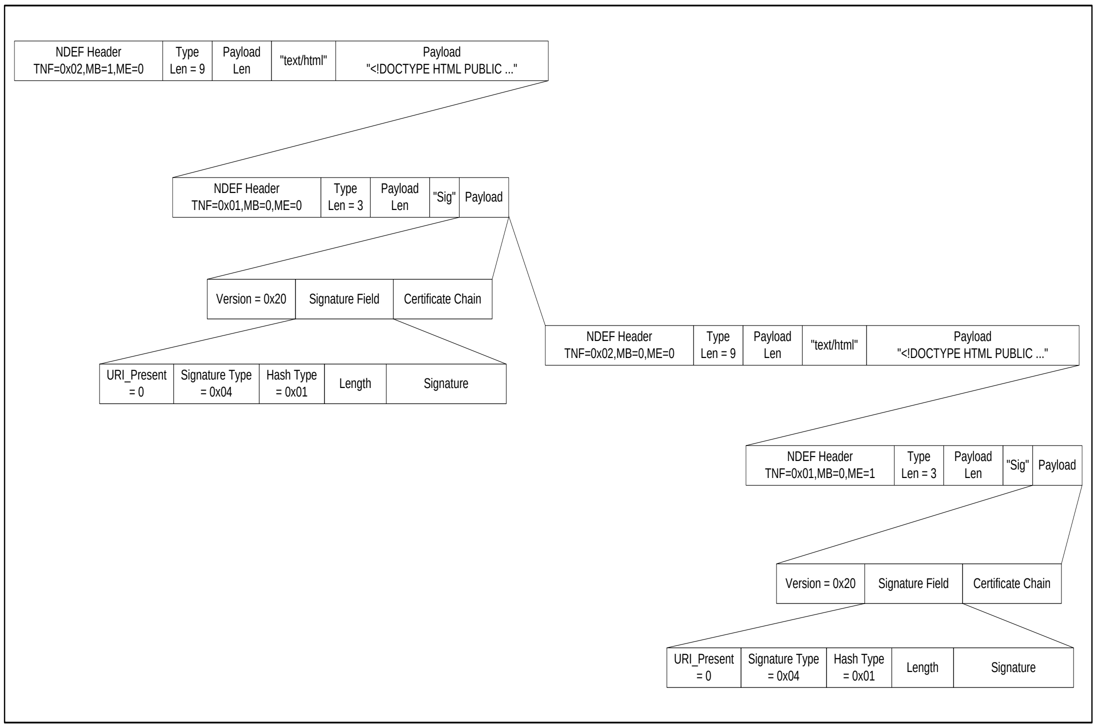

Figure 5: Signed MIME Record Example

E.1.3 Example 3

Figure 6 shows the format of a simple Smart Poster NDEF record, where the payload of the Smart Poster contains a Text record and a URI record before a signature is applied.

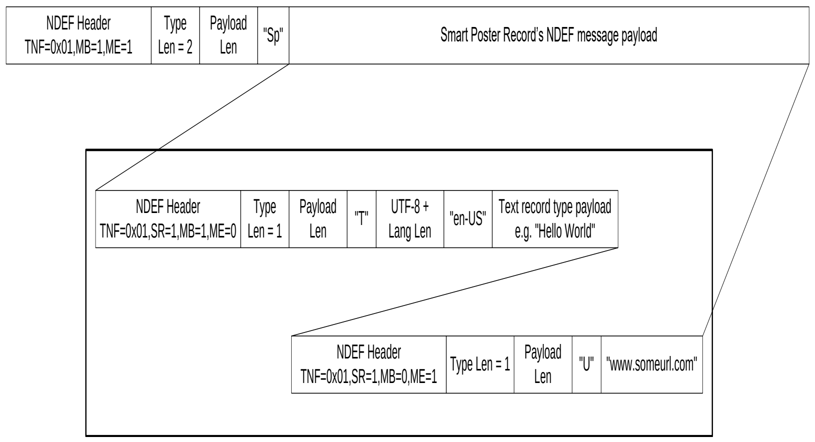
NDEF Header TNF=0x01,MB=1,ME=1
Type Len = 2
Payload Len
"Sp"
Smart Poster Record's NDEF message payload
NDEF Header TNF=0x01,SR=1,MB=1,ME=0
Type Len = 1
Payload Len
"T"
UTF-8 + Lang Len
"en-US"
Text record type payload e.g. "Hello World"
NDEF Header TNF=0x01,SR=1,MB=0,ME=1
Type Len = 1
Payload Len
"U"
"www.someurl.com"

Figure 6: Unsigned Smart Poster Record Example

Figure 7 shows the same NDEF message, but with a signature following the Smart Poster record that signs the whole Smart Poster record.

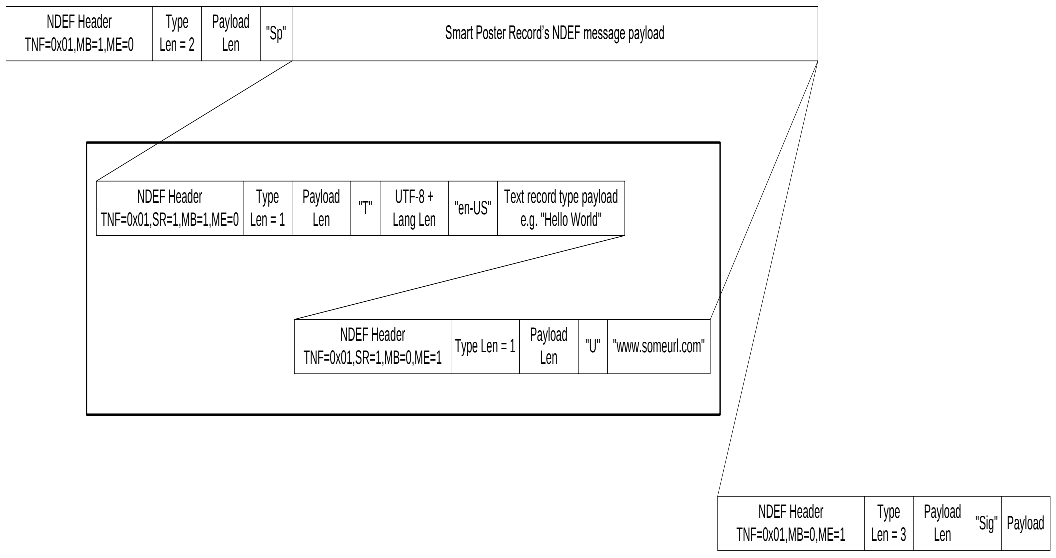
NDEF Header TNF=0x01,MB=1,ME=0
Type Len = 2
Payload Len
"Sp"
Smart Poster Record's NDEF message payload
NDEF Header TNF=0x01,SR=1,MB=1,ME=0
Type Len = 1
Payload Len
"T"
UTF-8 + Lang Len
"en-US"
Text record type payload e.g. "Hello World"
NDEF Header TNF=0x01,SR=1,MB=0,ME=1
Type Len = 1
Payload Len
"U"
"www.someurl.com"
NDEF Header TNF=0x01,MB=0,ME=1
Type Len = 3
Payload Len
"Sig"
Payload

Figure 7: Signed Smart Poster Record Example

E.2 Other Usage

The examples in this section violate the recommended practices in Appendix C, but might be useful in some special applications.

E.2.1 Example 4

Figure 8 shows the format of an NDEF message where there is a Text record ("Hello world") and a URI to a website before a signature is applied.

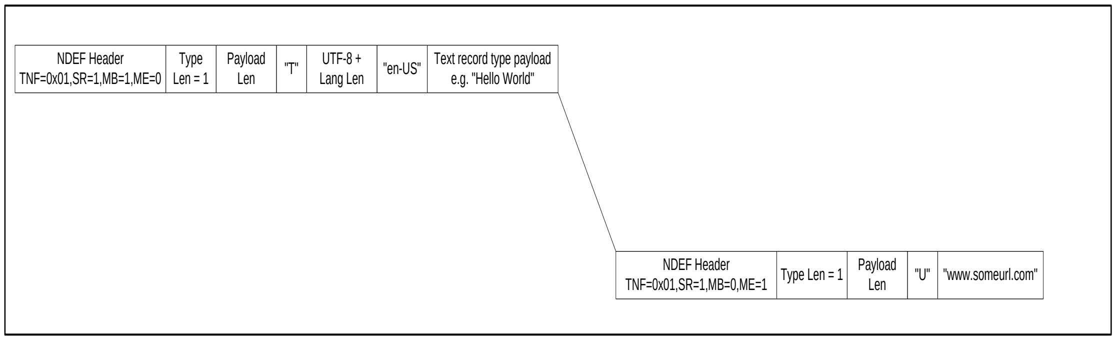
NDEF Header TNF=0x01,SR=1,MB=1,ME=0
Type Len = 1
Payload Len
"T"
UTF-8 + Lang Len
"en-US"
Text record type payload e.g. "Hello World"
NDEF Header TNF=0x01,SR=1,MB=0,ME=1
Type Len = 1
Payload Len
"U"
"www.someurl.com"

Figure 8: Unsigned Record Example

Figure 9 shows the same NDEF message, but with a signature start marker between the Text record and the URI record and a signature following the URI record. Only the URI record is included into the signature. Note that Appendix D.2 recommends not mixing signed and unsigned NDEF records.

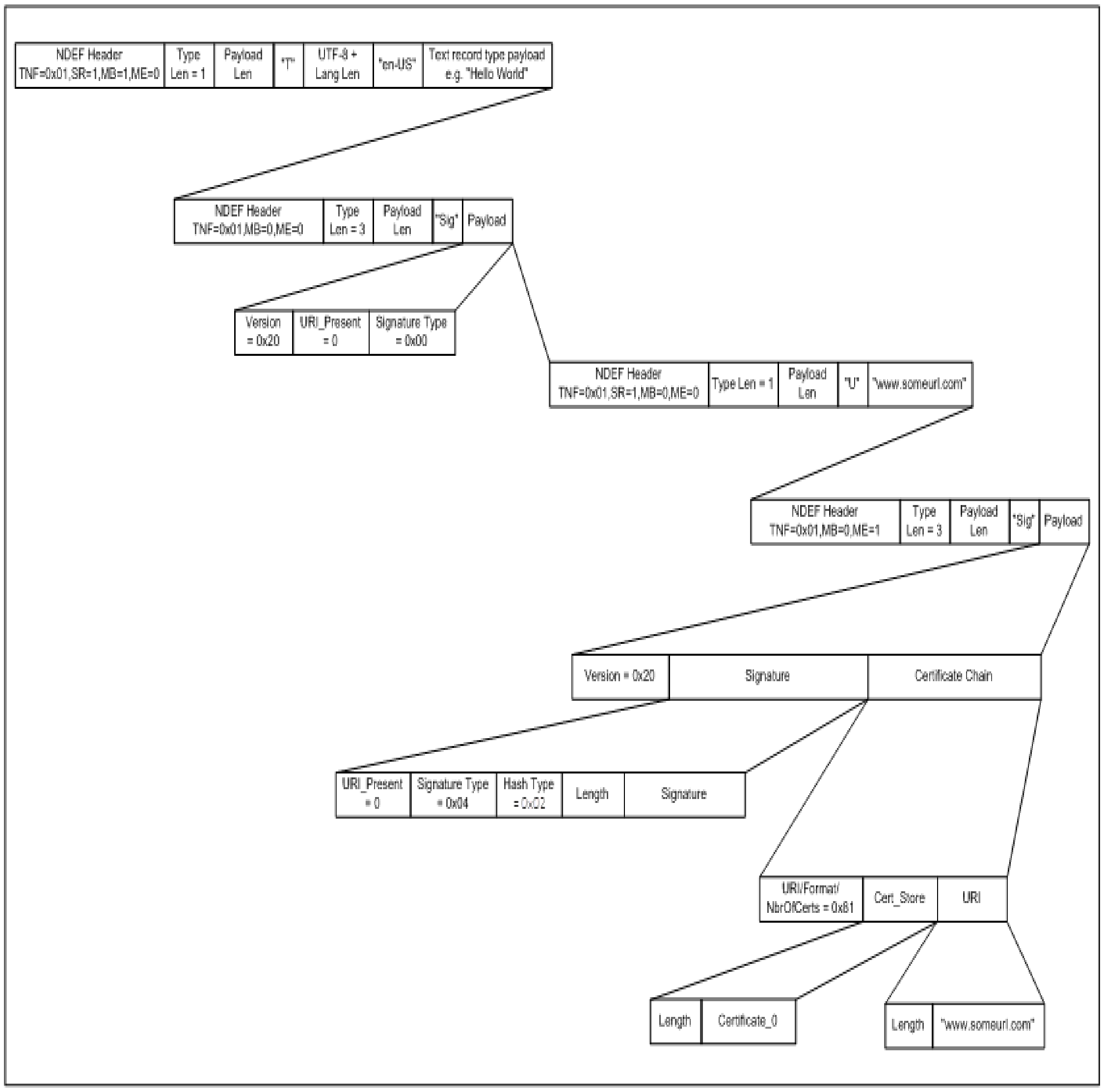

Figure 9: Signed Record Example

The Signature record contains a type 4 signature (ECDSA - P-192), a type 2 hash (SHA-256), and a type 0 (X.509) certificate chain; the certificate chain includes a URI.

E.2.2 Example 5

Figure 10 shows the format of a simple Smart Poster NDEF record, in which the payload of the Smart Poster contains a Text record and a URI record before a signature is applied.

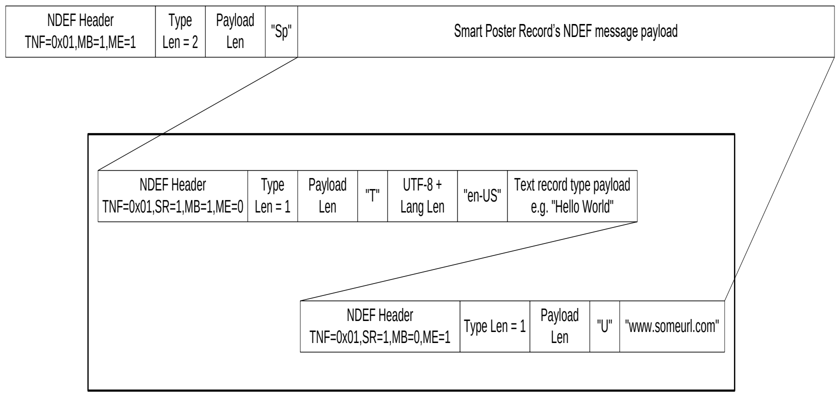
NDEF Header TNF=0x01,MB=1,ME=1
Type Len = 2
Payload Len
"Sp"
Smart Poster Record's NDEF message payload
NDEF Header TNF=0x01,SR=1,MB=1,ME=0
Type Len = 1
Payload Len
"T"
UTF-8 + Lang Len
"en-US"
Text record type payload e.g. "Hello World"
NDEF Header TNF=0x01,SR=1,MB=0,ME=1
Type Len = 1
Payload Len
"U"
"www.someurl.com"

Figure 10: Unsigned Smart Poster Record Example

Figure 11 shows the same NDEF message, but with a signature following the URI record in the nested NDEF message that signs the Text record and the URI record. When a Signature Record, along with other records, is nested in the payload of a record, such as shown in the Smart Poster example below, then the parent record (the Smart Poster) header, etc., is not included in the signature. Note that signing only the nested message of a Smart Poster record is not recommended (see Appendix D.3).

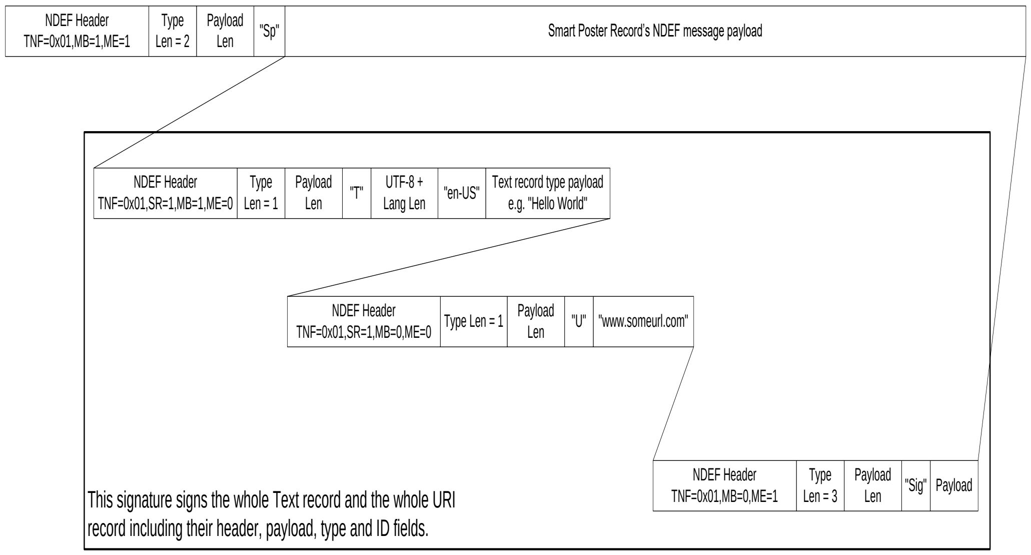
NDEF Header TNF=0x01,MB=1,ME=1
Type Len = 2
Payload Len
"Sp"
Smart Poster Record's NDEF message payload
NDEF Header TNF=0x01,SR=1,MB=1,ME=0
Type Len = 1
Payload Len
"T"
UTF-8 + Lang Len
"en-US"
Text record type payload e.g. "Hello World"
NDEF Header TNF=0x01,SR=1,MB=0,ME=0
Type Len = 1
Payload Len
"U"
"www.someurl.com"
NDEF Header TNF=0x01,MB=0,ME=1
Type Len = 3
Payload Len
"Sig"
Payload
This signature signs the whole Text record and the whole URI record including their header, payload, type and ID fields.

Figure 11: Signed Smart Poster Record Example

F. Revision History

Table 11 outlines the revision history of the Signature Record Type Definition Technical Specification.

Table 11: Revision History

Document Name	Revision and Release Date	Status	Change Notice	Supersedes
Signature Record Type Definition	Version 1.0, November 2010	Final		
Signature Record Type Definition	Version 2.0, November 2014	Final	Updates to version 1.0	Version 1.0, November 2010
Signature Record Type Definition	Version 2.0, December 2017	Final	Editorial corrections / alignments	Version 2.0, November 2014
Signature Record Type Definition	Version 2.0, February 2020	Final	Editorial update, including License page revision.	Version 2.0, December 2017