ISO/IEC TC JTC1/SC 17

Date: 2025-07-25

ISO/IEC 18013-5:xxxx(E)

Secretariat: BSI

Personal identification — ISO-compliant driving licence — Part 5: Mobile driving licence (mDL) application

Identification des personnes — Permis de conduire conforme à l'ISO —Partie 5: Application pour permis de conduire sur dispositif mobile (mDL)

This is the CD ballot resolution document of ISO/IEC 18013-5 second edition. After a 2-week review this document can go to the next phase for DIS.

Items to add during 2 week review :

- Intro to second edition
- Add examples for:
  - Example for extendend requests, including UniqueDocSetRequired
  - What the statuslist and identifierlist CDDL looks like in the context of this document
  - Negotiated handover
- Verify editors notes and comments

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 17, *Cards and security devices for personal identification*.

A list of all parts in the ISO/IEC 18013 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

The ISO/IEC 18013 series establishes guidelines for the design format and data content of an ISO-compliant driving licence (IDL) with regard to human-readable features (ISO/IEC 18013-1), ISO machine-readable technologies (ISO/IEC 18013-2), access control, authentication and integrity validation (ISO/IEC 18013-3), and associated test methods (ISO/IEC 18013-4). It creates a common basis for international use and mutual recognition of the IDL without impeding individual countries/states in applying their privacy rules and national/community/regional motor vehicle authorities in taking care of their specific needs.

This document describes interface and related requirements to facilitate ISO-compliant driving licence (IDL) functionality on a mobile device. The requirements are specifically intended to enable verifiers not affiliated with or associated with the issuing authority to gain access to and authenticate the information. In addition, the requirements allow the holder of the driving licence to decide what information to return to a verifier. Other characteristics include the ability to update information frequently, and to authenticate information at a high level of confidence.

A mobile document conforming to this document primarily conveys the driving privileges associated with a person. It does so by providing means to associate the person presenting the mobile document with the mobile document itself. However, the transaction and security mechanisms in this document have been designed to support other types of mobile documents, specifically including identification documents. Consequently the mechanisms in this document can be used for any type of mobile identification document, regardless of the additional attributes the mobile document may convey. The details of the data elements to be used by other mobile documents are left to the respective issuing authority and are not within the scope of this document.

Personal identification — ISO-compliant driving licence — Part 5: Mobile driving licence (mDL) application

1 Scope

This document establishes interface specifications for the implementation of a driving licence in association with a mobile device. This document specifies the interface between the mobile driving licence (mDL) and mDL reader and the interface between the mDL reader and the issuing authority infrastructure. This document also enables parties other than the issuing authority (e.g. other issuing authorities, or mDL verifiers in other countries) to:

— use a machine to obtain the mDL data;

— tie the mDL to the mDL holder;

— authenticate the origin of the mDL data;

— verify the integrity of the mDL data.

The following items are out of scope for this document:

— how mDL holder consent to share data is obtained;

— requirements on storage of mDL data and mDL private keys.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 3166-1, *Codes for the representation of names of countries and their subdivisions — Part 1: Country code*

ISO 3166-2:2020, *Codes for the representation of names of countries and their subdivisions — Part 2: Country subdivision code*

ISO/IEC 5218, *Information technology — Codes for the representation of human sexes*

ISO/IEC 7816-4:2020, *Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange*

ISO/IEC 8859-1, *Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1*

ISO/IEC 18004, *Information technology — Automatic identification and data capture techniques — QR Code bar code symbology specification*

ISO/IEC 18013-1:2018, *Information technology — Personal identification — ISO-compliant driving licence — Part 1: Physical characteristics and basic data set*

ISO/IEC 18013-2:2020, *Personal identification — ISO-compliant driving licence — Part 2: Machine-readable technologies*

ISO/IEC 19785-3:2020, *Information technology — Common Biometric Exchange Formats Framework — Part 3: Patron format specifications*

ISO/IEC 26131:2024, *Information technology — OpenID Connect — OpenID Connect Core 1.0 incorporating errata set 2*

ISO/IEC 26132:2024 — *Information technology — OpenID Connect — OpenID Connect Discovery 1.0 incorporating errata set 2*

BSI TR-03111, *Elliptic Curve Cryptography, Version 2.10, June 2018*

NFC Forum, *Connection Handover (CH) Technical Specification, Version 1.5*

NIST SP 800-186, *Recommendations for Discrete Logarithm-based Cryptography, February 2023*

NIST SP 800-38D, *M. Dworkin, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, November 2007*

RFC 4122, *P. Leach et al., A Universally Unique IDentifier (UUID) URN Namespace, July 2005*

RFC 4648, *S. Josefsson, The Base16, Base32, and Base64 Data Encodings, October 2006*

RFC 5246, *T. Dierks et al., The Transport Layer Security (TLS) Protocol Version 1.2, August 2008*

RFC 5280, *D. Cooper et al., Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, May 2008*

RFC 5639, *M. Lochter et al., Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, March 2010*

RFC 5869, *H. Krawczyk, HMAC-based Extract-and-Expand Key Derivation Function (HKDF), May 2010*

RFC 6066, *D. Eastlake 3rd, Transport Layer Security (TLS) Extensions: Extension Definitions, January 2011*

RFC 7515, *J. Bradley et al., JSON Web Signature (JWS), May 2015*

RFC 7518, *M. Jones et al., JSON Web Algorithms (JWA), May 2015*

RFC 7519, *J. Bradley et al., JSON Web Token (JWT), May 2015*

RFC 7748, *A. Langley et al., Elliptic Curves for Security, January 2016*

RFC 7905, *A. Langley et al., ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS), June 2016*

RFC 8032, *S. Josefsson et al., Edwards-Curve Digital Signature Algorithm (EdDSA), January 2017*

RFC 8252, *W. Denniss et al., Oauth 2.0 for Native Apps, October 2017*

RFC 8259, *T. Bray, The JavaScript Object Notation (JSON) Data Interchange Format, December 2017*

RFC 8410, *S. Josefsson et al., Algorithm Identifiers for Ed25519, Ed448, X25519, and X448 for Use in the Internet X.509 Public Key Infrastructure, August 2018*

RFC 8422, *Y. Nir et al., Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier, August 2018*

RFC 8943, *M. Jones et al., Concise Binary Object Representation (CBOR) Tags for Date, November 2020*

RFC 8949, *C. Bormann et al., Concise Binary Object Representation (CBOR), December 2020*

RFC 9052, *J. Schaad et al., CBOR Object Signing and Encryption (COSE): Structures and Process, August 2022*

RFC 9053, *J. Schaad et al. CBOR Object Signing and Encryption (COSE): Initial Algorithms, August 2022*

RFC 9360, *CBOR Object Signing and Encryption (COSE): Headers Parameters for Carrying and Referencing X.509 certificates, December 2023*

RFC xx, Token Status List, *draft-ietf-oauth-status-list-12, July 2025*

Wi-Fi Alliance, *Neighbor Awareness Networking Specification, Version 3.1*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
mobile device
portable computing device that at least:

— has a small form factor such that it can easily be carried by a single individual;

— is designed to operate, transmit and receive information without a wired connection;

— possesses local, nonremovable or removable data storage;

— includes a self-contained power source;

— includes a display;

— includes a means for the holder of the portable computing device to interact with the device

Note 1 to entry: Adapted from NIST SP 800-157.

3.2
mdoc
document or application that resides on a *mobile device* (3.1) or requires a mobile device as part of the process to gain access to the document or application

3.3
mdoc reader
device that can retrieve *mdoc* (3.2) data for verification purposes

3.4

mdoc holder
individual to whom an *mdoc* (3.2) is issued

3.5
mdoc verifier
person or organization using and/or controlling an *mdoc reader* (3.3) to verify an *mdoc* (3.2)

3.6
mDL
driving licence that fulfils at least the same function as an IDL but, instead of being paper or plastic based, is an *mdoc* (3.2)

Note 1 to entry: ISO-compliant driving licence (IDL) is defined in ISO/IEC 18013-1.

3.7
mDL reader
mdoc reader (3.3) that can retrieve *mDL* (3.6) data

3.8
mDL holder
individual to whom an *mDL* (3.6) is issued, i.e. legitimate holder of the driving privileges reflected on an mDL

3.9
mDL verifier
person or organization using and/or controlling an *mDL reader* (3.7) to verify an *mDL* (3.6)

3.10
licensing authority
authorized agent organisation that issues a driving licence

EXAMPLE National, federal, state, provincial, regional, territorial, or local Ministry of Transport, Department of Motor Vehicles, or Police Agency.

[SOURCE: ISO/IEC 18013-1:2018, 3.15]

3.11
issuing country
country which issued the driving licence or within which the *licensing authority* (3.10) is located

[SOURCE: ISO/IEC 18013-1:2018, 3.12, modified — The words "according to Annex F" have been removed.]

3.12
issuing authority
licensing authority (3.10), or *issuing country* (3.11) if separate licensing authorities have not been authorized

[SOURCE: ISO/IEC 18013-1:2018, 3.11]

3.13
issuing authority infrastructure
infrastructure under control of the *issuing authority* (3.12)

3.14

issuing authority CA
certificate authority operated by or on behalf of an *issuing authority* (3.12)

3.15
device retrieval
method of data retrieval exclusively using the interface between the *mdoc* (3.2) and the *mdoc reader* (3.3)

3.18
PCD mode
mode in which an NFC-enabled *mobile device* (3.1) operates as a PCD

[SOURCE: ISO/IEC 14443-3:2018, 3.7, modified — The words “a PXD” have been replaced with “an NFC-enabled mobile device”.]

3.19
PICC mode
mode in which an NFC-enabled *mobile device* (3.1) operates as a PICC

[SOURCE: ISO/IEC 14443-3:2018, 3.8, modified — The words “a PXD” have been replaced with “an NFC-enabled mobile device”.]

4 Abbreviated terms

AES	advanced encryption standard
APDU	application protocol data unit
BLE	Bluetooth® low energy
BT SIG	Bluetooth special interest group
CA	certificate authority
CBOR	concise binary object representation
CDDL	concise data definition language
COSE	CBOR object signing and encryption
CSPRNG	cryptographically secure pseudo-random number generator
CRL	certificate revocation list
DER	distinguished encoding rules
DS	document signer
ECDH	elliptic curve Diffie-Hellman key agreement
ECDSA	elliptic curve digital signature algorithm
EdDSA	Edwards-curve digital signature algorithm
GATT	generic attribute profile
HKDF	HMAC-based extract-and-expand key derivation function
HMAC	hash-based MAC
IA	issuing authority
IACA	issuing authority certificate authority
IANA	internet assigned number authority

IDL	ISO-compliant driving licence
IKM	input keying material
JSON	JavaScript object notation
JWK	JSON web key
JWS	JSON web signature
JWT	JSON web token
KDF	key derivation function
MAC	message authentication code
MITM	man-in-the-middle attack
MSO	mobile security object
MTU	maximum transmission unit
NDEF	NFC data exchange format
NFC	near field communication
OCSP	online certificate status protocol
OID	object identifier
OIDC	OpenID connect
PCD	proximity coupling device
PICC	proximity card or object
PKI	public key infrastructure
RF	radiofrequency
RFU	reserved for future use
SHA	secure hash algorithm
TLS	transport layer security
URI	uniform resource identifier
URL	uniform resource locator
UTC	coordinated universal time
UUID	universally unique identifier
VICAL	verified issuer certificate authority list

5 Conformance requirement

An mDL is in conformance with this document if it meets all the requirements specified directly or by reference herein. Conformance with ISO/IEC 18013-1, ISO/IEC 18013-2, ISO/IEC 18013-3, and ISO/IEC 18013-4 is not required for conformance with this document, except for those clauses normatively referenced in this document.

An mDL reader is in conformance with this document if it meets all the requirements specified directly or referenced herein.

An issuing authority infrastructure is in conformance with this document if it meets all the requirements specified directly or referenced herein.

Editor’s note: we should add a statement that this second edition has been designed such that an implementation compliant with ISO/IEC 18013-5:2021 is also compliant with this edition and can therefore interact with a device compliant with this edition and vice versa.

6 Intro to second edition

In this second edition of the document the following major features have been added;

— Additionial information to describe the requested information;

— mDL revocation mechanism;

— Zero knowledge proof support.

In addition, a number of smaller improvements have been made;

— Support sending the request during NFC handover

— BLE L2CAP mode improvements

— Editorial re-ordering of the document for readability

— Reader Identity Certificate Authority list structure

Editor’s note: still to be added is to explain the decisions of removing server retrieval from this edition.

7 mDL overview

7.1 Interfaces

Figure 1 shows the different mDL interfaces.

— Interface 1 in Figure 1 is the interface between the mDL and the mDL reader. This is the primary interface specified in this document. This interface can be used for connection setup and for the device retrieval method.

— Interface 2 in Figure 1 is an interface between the issuing authority infrastructure and an mDL reader. This interface is out of scope for this document.

— Interface 3 in Figure 1 is the interface between the issuing authority infrastructure and the mDL. This interface is out of scope for this document.

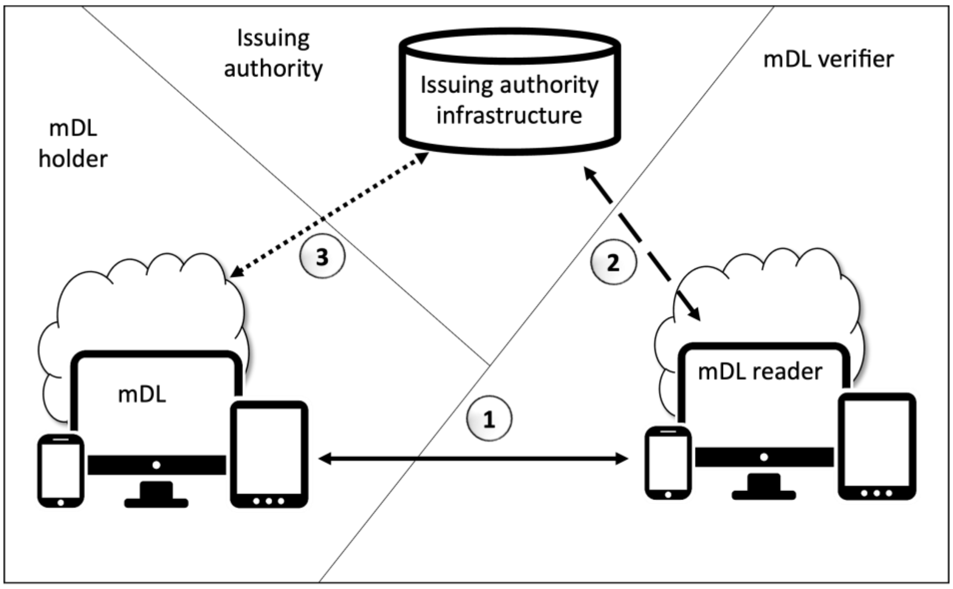

Figure 1 — mDL interfaces

7.2 Functional requirements

The objectives underlying the requirements in this document include at least the following.

a) An mDL verifier together with an mDL reader shall be able to request, receive and verify the integrity and authenticity of an mDL whether online connectivity is present or not for either the mDL or mDL reader.

b) An mDL verifier not associated with the issuing authority shall be able to verify the integrity and authenticity of an mDL.

c) An mDL verifier shall be enabled to confirm the binding between the person presenting the mDL and the mDL holder.

d) The interface between the mDL and the mDL reader shall support the selective disclosure of mDL data to an mDL reader.

How mDL holder consent to share data is obtained and requirements on storage of mDL data and mDL private keys are out of scope of this document, however Annex E contains considerations for consent and key storage.

7.3 Technical requirements

The mDL and mDL reader are implemented as a profile of an mdoc and mdoc reader respectively. The requirements for mdoc and mdoc reader are defined in Clause 8, Clause 9, Clause 10, Clause 11 and Clause 12.

The requirements for the mDL, mDL reader and issuing authority are specified in Clause 13. These include the requirements for the mDL data model, the engagement and data retrieval methods requirements and the security mechanism requirements.

NOTE The concepts for mdoc and mdoc reader used in this document have been designed so that other mobile credentials, e.g. mobile identity or other credentials that substitute the mDL data elements with a different set of data elements, can also make use of the engagement and retrieval protocols described in this document. Specifically, the mdoc data model is based on elements with unique identifiers within a namespace. The number of elements can vary, and the model is indifferent to the value and data format of each element. As such the data model is generic and can apply to any kind of document.

8 Mdoc mechanisms

8.1 Mdoc data model

The mdoc data model defines a structured way to represent the data in a mobile document. An mdoc is identified by a doctype. The data elements within the mdoc are identified through a combination of namespace and element identifier. These data elements are then cryptographically secured using a mobile secure object (MSO).

The `DocType` parameter is a text string that identifies the document type of the mdoc or the requested document type by the mdoc reader. There is no requirement for the `DocType` format. An approach to avoid collisions is to use the following general format: [Reverse Domain].[Domain Specific Extension].

The `NameSpace` parameter is a text string that provides the namespace within which the data elements requested by the mdoc reader or returned by the mdoc are defined. A document may contain data elements from multiple namespaces. The meaning of data elements is dependent on their namespace. There is no requirement for the `NameSpace` format. An approach to avoid collisions is to use the following general format: [Reverse Domain].[Domain Specific Extension].

A data element has a `DataElementIdentifier` that is a text string that identifies the data element and a `DataElementValue` that contains the value of the data element.

Note A data element is unique defined by the namespace it belongs to and its identifier. Therefore, two data elements that have the same data element identifier but belong to a different namespace are fully distinct data elements.

The data elements and related information is protected using the mobile security object (MSO) which also contains the mdoc public key. The mechanism used by the MSO to protect the data elements (using salted hashes) allows the data elements in the mobile document to be returned individually. Therefore, when performing a transaction, only the data elements necessary for that transaction are returned to the mdoc reader. This is further described in 12.3.

The resulting mdoc data model is illustrated in figure 2.

Since the `Doctype`, `Namespace`, and `DataElementIdentifier` are UTF-8 encoded text strings, the equality of their values is determined by an exact match of the string value. This also means that their values are case-sensitive.
`Doctype`, `NameSpace` and `DataElementIdentifier` shall not be empty strings.

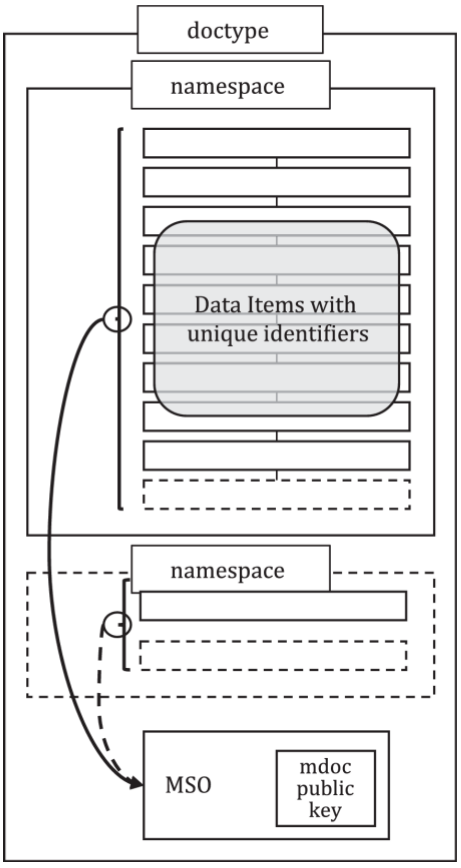

Figure 2 — mdoc data model

8.2 Data exchange

8.2.1 Overview

Data exchange is divided into three phases: the initialization phase, the device engagement phase, and the data retrieval phase. After initialization between the mdoc and the mdoc reader follows device engagement, followed by exchange of data between the mdoc and the mdoc reader.

NOTE 1 There is no requirement for any device involved in the transaction to be connected to the internet.

NOTE 2 The transaction has been designed such that it is not necessary for the mDL holder to physically hand over the mobile device to the mDL verifier.

The request and response commands are applicable to any kind of mobile document based on the mdoc data model. Moreover, these commands can work with mdocs that contain multiple documents. This is because elements from different documents can be requested, and the response can include multiple documents from the same or different document type.

8.2.2 Initialization

During initialization, the mdoc is activated. Activation is done by the mdoc holder or triggered by an mdoc reader using NFC. Simultaneously, the mdoc reader is activated. No requirements are specified for this phase.

NOTE It is important to avoid unauthorized access by an mdoc reader if mdoc activation is triggered by NFC.

8.2.3 Device engagement

During device engagement, information required to setup and secure data retrieval is exchanged between the mdoc and the mdoc reader. The device engagement information, described in 9.1, is transferred using one of the transmission technologies.

Transmission technologies available to transfer the device engagement dat a are as follows:

- NFC (9.2)
- QR code (9.3)

8.2.4 Data retrieval architecture

Figure 3 shows the different interfaces, transmission technologies and the flow of the messages for device retrieval.

During data retrieval, the mdoc and mdoc reader communicate using mdoc request and mdoc response messages encoded with CBOR. These messages are transported using a transmission technology. The transmission technologies are agnostic to the information that is transferred.

After device engagement, the mdoc reader asks for data using the mdoc request as defined in 10.2. The mDL sends an mdoc response as defined in 10.3.

The different transmission technologies are described in Clause 11. The transmission technologies defined in this document use session encryption as defined in 12.2 to provide application layer encryption of the request and response messages that is independent of the transmission technology used.

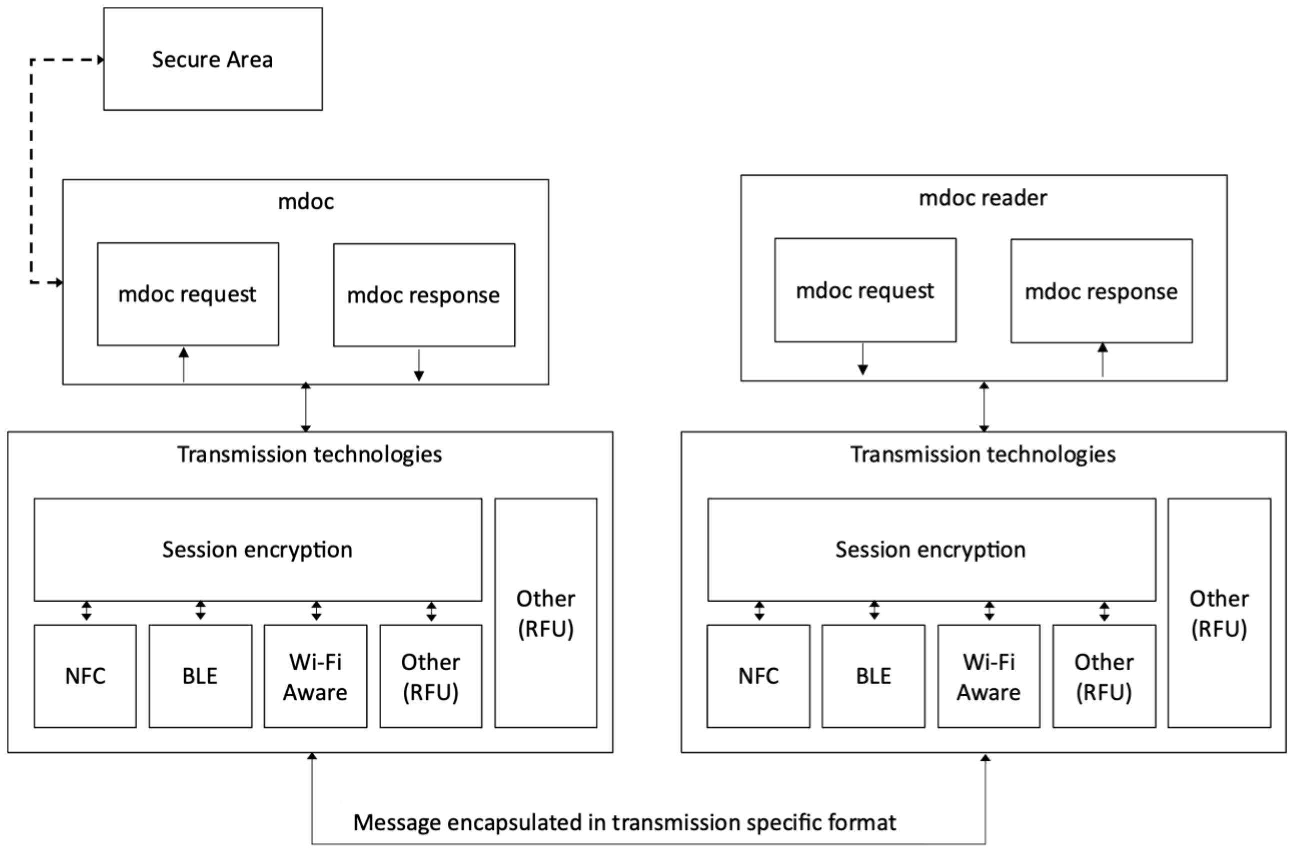

Figure 3 — Data retrieval architecture

NOTE 1 The secure area as present in Figure 3 indicates an area that provides additional protection of sensitive mdoc related data. Security requirements regarding storage of credential information are outside the scope of this document. This includes the mdoc private key and the mdoc reader private key, if mdoc reader authentication is supported by the mdoc reader. It is the responsibility of the issuing authority to ensure that all data stored on the mdoc is stored securely.

NOTE 2 Implementation possibilities for a secure area are non-exhaustively listed in E.5.

8.2.5 Transmission technologies

The following transmission methods are available to transmit the mdoc request and mdoc response for device retrieval:

- BLE (11.1)
- NFC (11.2)
- Wi-Fi Aware (11.3)

See Annex D for examples of data structures.

NOTE 1 If QR is used for device engagement and the mDL reader chooses to use NFC for data transfer, then there is no mechanism available for the mDL reader to indicate the choice for NFC data transfer to the mDL. It is possible that the mDL holder is not aware that the mDL needs to interface with the mDL reader using NFC. On the contrary, if NFC is used for device engagement, this problem does not exist.

NOTE 2 Due to the limited data transfer rate for NFC, if a large amount of data is required for a transaction, it is possible that it is neither practical nor reasonable to have an mDL holder hold the device within the RF range of the mDL reader for the duration of the transaction. Furthermore, due to the loss of signal when a device leaves the RF field, any mDL holder interactions with the mDL causing the mDL to leave the RF field require a new transaction to be initiated. This can be avoided by having all mDL holder interactions with the mDL done while the mDL stays in the field or if mDL does not require any mDL holder interactions while it is in the RF field.

8.3 Encoding of data structures and data elements

In this document CDDL (Concise Data Definition Language) as specified in RFC 8610 is used to express CBOR encoded data structures.

CBOR structures shall be encoded according to RFC 8949. The following requirements apply when encoding and decoding CBOR structures:

- The first two restrictions of core deterministic encoding as defined in section 4.2.1 of RFC 8949 shall be implemented. These are the requirements related to preferred serialization and indefinite-length items.
- The last restriction of core deterministic encoding as defined in section 4.2.1 of RFC 8949 should be implemented. This is the requirement related to sorting of keys in maps.
- When encountering an invalid CBOR structure, a parser is recommended to stop parsing and return an error. Multiple identical keys in a map is considered an invalid CBOR structure.

Because deterministic encoding of sorting keys in maps is not required, all CBOR maps that are used in a cryptographic operation are communicated in a tagged CBOR bytestring. For any cryptographic operation, an mdoc, mdoc reader or issuing authority infrastructure shall use these bytestrings as they were sent or received, without attempting to re-create them from the underlying maps.

EXAMPLE A data structure `DataItem` that is to be used in a cryptographic operation is communicated in a structure `DataItemBytes,` specified as follows:

```
DataItemBytes = #6.24(bstr .cbor DataItem)
```

The CDDL in this example is defined in RFC 8610, section 3.6 and expresses a tagged data item (major type 6). As specified in RFC 8949, section 3.4.5.1, tag value 24 indicates that the content of the CBOR bstr following the tag is itself a CBOR data item. The .cbor control operator indicates that this data item is in fact a `DataItem`.

Whenever data structures in this document use a version element that is encoded as a string, their contents follow the format of 'major version number'.'minor version number'. A major version number shall be incremented by 1 when any backwards incompatible changes are introduced. In a future version of this document. A minor version number shall be incremented by 1 when new, but backwards compatible functionality is introduced. A minor version number shall be reset to 0 if the major version number is incremented. An mdoc, mdoc reader or issuing authority infrastructure shall not give an error and continue a transaction if it receives a data structure having a known major version number but with an unknown minor version number. The `Version` element used throughout this specification has the following CDDL definition:

```
Version = tstr
```

Structures defined in this document use `RFU` and `Ext` to indicate extension points. They have the following CDDL definition:

```
RFU = any
Ext = any
```

`RFU` is reserved for future use by ISO/IEC JTC1/SC17. `Ext` may be used for application-specific extensions. This means that application-specific extensions cannot be used at places where this is not explicitly permitted by the definition of the structure. When `Ext` is used in a map that uses strings as keys, a collision-resistant key using the reverse domain name approach should be used. An mdoc, mdoc reader or issuing authority infrastructure shall not give an error when encountering an `RFU` or unknown `Ext` extension point.

Throughout this specification, as a general rule the keys in maps use a lowercase letter, while the name of the related structure uses a capital letter.

The CDDL definitions for `DocType`, `NameSpace`, `DataElementIdentifier` and `DataElementValue` are common across different data retrieval methods and the applicable security mechanisms. The following CDDL definitions are used in this document:

```
DocType = tstr
NameSpace = tstr
DataElementIdentifier = tstr
DataElementValue = any
```

9 Device engagement

9.1 Device engagement information

The device engagement structure contains information to perform device engagement. The device engagement structure shall be CBOR encoded and formatted as follows:

```
DeviceEngagement =
{
    0: Version,
    1: Security,
    ? 2: DeviceRetrievalMethods,
    ? 5: OriginInfos,
    ? 6: Capabilities,
    * uint => RFU,
    * nint => Ext
}

Security = [
    int,               ; Cipher suite identifier
    EDeviceKeyBytes
]

DeviceRetrievalMethods = [
    + DeviceRetrievalMethod
]

ServerRetrievalMethods = {
    ?"webApi": WebApi,
    ?"oidc": Oidc
    tstr => RFU
}

DeviceRetrievalMethod = [
    uint,   ; Type
    uint,   ; Version
    RetrievalOptions
]


OriginInfos = [
    * OriginInfo
]

OriginInfo = RFU

Capabilities = {
    ? 2: HandoverSessionEstablishmentSupport,
    ? 3: ReaderAuthAllSupport
    ? 4: ExtendedRequestSupport
    * int => RFU
}

HandoverSessionEstablishmentSupport = bool
ReaderAuthAllSupport = bool
ExtendedRequestSupport = bool


RetrievalOptions = WifiOptions / BleOptions / NfcOptions / RFU
```

The device engagement structure contains the following key-value pairs.

0. `Version`: the version of the device engagement structure. When key 5 or 6 are present, the version shall be “1.1”, otherwise the version shall be “1.0”.

1. `Security`: an array that contains two mandatory elements. The first element is the cipher suite identifier, defined in 12.6.2. The second element is `EdeviceKeyBytes`, defined in 12.2.4.

2. `DeviceRetrievalMethods`: an array that shall contain one or more `DeviceRetrievalMethod` arrays when performing device engagement using the QR code. When using NFC to perform device engagement, the `DeviceRetrievalMethods` array shall be absent,

because the data retrieval methods supported by the mdoc are specified in the Alternative Carrier Records, as specified in 9.2. `DeviceRetrievalMethods` lists the device retrieval methods supported by the mdoc. A `DeviceRetrievalMethod` array holds two mandatory values (type and version). The first element defines the type and the second element the version for the transfer method. A `DeviceRetrievalMethod` with a particular combination of type and version shall only be present once. The `RetrievalOptions` element contains extra info for each data retrieval method. The values for the different retrieval options are defined in 9.3.

5. `OriginInfos`: The `OriginInfos` structure provides information about the interface used to receive and deliver the engagement structure. If `Capabilities` is present, `OriginInfos` shall be present. When the device engagement structure is used with one of the flows as described in clause 6.3.2.1, the `OriginInfos` array may be empty.

6. `Capabilities`: Contains a map of capabilities the mdoc supports. This document defines three capabilities: `HandoverSessionEstablishmentSupport`, `ReaderAuthAllSupport` and `ExtendedRequestSupport` as defined below.

Note 1 The requirement for `OriginInfos` to be present (even if empty) when `Capabilities` is present is to ensure compatibility with the definition of the `DeviceEngagement` structure in ISO/IEC 18013-7

Note 2 The first edition of this document, ISO/IEC 18013-5:2021, defines key 3 for ServerRetrievalMethods, which is not included in this edition of ISO/IEC 18013-5 and key 4 for ProtocolInfo as RFU. Since both these keys are positive integers, they can be considered RFU for any implementation implementing this edition of ISO/IEC 18013-5 and they have therefore not been explicitly included in the CDDL definition of `DeviceEngagement`.

The `HandoverSessionEstablishmentSupport` element in the `Capabilities` structure may be used by the mdoc to indicate whether it supports receiving the `SessionEstablishment` message during Negotiated Handover as defined in 9.2.3. If `HandoverSessionEstablishmentSupport` is present its value shall be `True`.

The `ReaderAuthAllSupport` element in the `Capabilities` structure may be used by the mdoc to indicate whether it supports receiving the ReaderAuthAll structure in the mdoc request as defined in 10.2.6. If `ReaderAuthAllSupport` is present its value shall be `True`.

The `ExtendedRequestSupport` element in the `Capabilities` structure can be used by the mdoc to indicate whether it supports parsing the parameters specified for the DocRequestInfo and DeviceRequestInfo. When it supports parsing `alternativeDataElements`, `issuerIdentifiers` and `useCases` the value shall be set to `True`. When it does not support all these 3 elements, its value should be set to `False`. The behavoir of an mdoc reader receiving this value is out of scope of this specification.

An example of a device engagement structure can be found in D.3.1.

9.2 Device engagement using NFC

9.2.1 General mechanism

Device engagement using NFC shall follow the Connection Handover protocol as defined by NFC Forum, *Connection Handover (CH) Technical Specification, Version 1.5*. Only Reader/Writer mode using the Type 4 Tag shall be used. The Connection Handover protocol shall be initiated by the mdoc reader. The mdoc reader shall take the role of Handover Requester. The mdoc shall be the NFC Tag Device and the mdoc reader shall be the NFC Reader Device. The mdoc shall use either Static Handover or Negotiated Handover.

When Static Handover is used, the Handover Select Message shall be retrieved by the mdoc reader from the mdoc in a Type 4 Tag and shall contain at least one Alternative Carrier Record. Each Alternative Carrier Record shall indicate a device retrieval method the mdoc supports. An mdoc reader shall select one of the transmission technologies from the ones provided in the Alternative Carrier Records.

When Negotiated Handover is used, the mdoc shall include the "urn:nfc:sn:handover" service in a Service Parameter record in the Initial NDEF message provided to the mdoc reader. The mdoc reader shall send a Handover Request Message to the mdoc after it has selected this service. The Handover Request Message shall contain an Alternative Carrier Record for each alternative carrier that is supported by the mdoc reader. The mdoc confirms the handover by providing a Handover Select Message containing exactly one selected alternative carrier.

NOTE Use of Negotiated Handover for device engagement allows negotiation of transfer methods. For BLE and Wi-Fi Aware, it additionally allows negotiation of keys used by the transmission layer. This can provide better user experience and security of data transmission.

For the BLE device retrieval transmission technology, the contents of the Alternative Carrier Record and Carrier Configuration Record(s) shall comply with 11.1.2.

For the NFC device retrieval transmission technology, the contents of the Alternative Carrier Record and Carrier Configuration Record(s) shall comply with 9.2.2.

For the Wi-Fi Aware device retrieval transmission technology, the contents of the Alternative Carrier Record and Carrier Configuration Record(s) shall comply with the Wi-Fi Alliance Neighbor Awareness Networking Specification, version 3.1, section 12.

The `DeviceEngagement` structure as defined in 9.1 shall be transferred from the mdoc to the mdoc reader as part of an auxiliary data record of the Handover Select Message with the type "iso.org:18013:deviceengagement" and the ID reference "mdoc", with type name format NFC Forum external type (0x04). For each ac record, "Auxiliary Data Reference" points to the NDEF record which contains the device engagement structure.

When Negotiated Handover is used, the mdoc reader may send a `ReaderEngagement` structure to the mdoc as part of an auxiliary data record of the Handover Request Message with the type "iso.org:18013:readerengagement" and the ID reference "mdocreader", with type name format NFC Forum external type (0x04), to provide information from the mdoc reader to the mdoc.

When Negotiated Handover is used, the mdoc reader should send the SessionEstablishment message during Negotiated Handover according to clause 9.2.3 if the mdoc indicated support for this in the Capabilities structure in the device engagement structure.

The reader engagement structure shall be CBOR encoded and formatted as follows:

```
ReaderEngagement =
{
    0: tstr,            ; Version
    * uint => RFU,
    * nint => Ext
}
```

The reader engagement structure contains the following key-value pairs:

0. `Version`: the version of the reader engagement structure. In the current version of this document its value shall be "1.0".

This document does not define any other key-value pairs for use in `ReaderEngagement`.

9.2.2 Alternative Carrier Record for device retrieval using NFC

The Alternative Carrier Record for the NFC device retrieval transmission technology shall reference the Carrier Configuration Record with the ID reference “nfc”.

The Carrier Configuration Record for NFC device retrieval transmission technology shall have the type “iso.org:18013:nfc” and the ID reference “nfc”. The binary content of the Carrier Configuration Record shall be encoded according to Table 1.

Table 1 — Binary content of the Carrier Configuration Record for NFC device retrieval

Field	Size (Octet)	Sub-Field	Value	Presence
Version	1		0x01	Mandatory
Maximum command data length	1	Length	See below	Conditional
	1	Data type	0x01	
	Variable	Maximum command data length	See 8.3.3.1.2	
Maximum response data length	1	Length	See below	Conditional
	1	Data type	0x02	
	Variable	Maximum response data length	See 8.3.3.1.2	

The value of the version field shall be the mdoc NFC Connection Handover Version encoded as an unsigned integer. This is a mandatory field.

For each field, the value of the length sub-field shall be the sum of the length of the other two sub-fields within that field, encoded as an unsigned integer.

The value of the maximum command data length field shall be the maximum length of command data field supported by the mdoc, encoded as an unsigned integer. This field is further defined in 11.2. This field is mandatory for the mdoc and shall not be used by the mdoc reader.

The value of the maximum response data length field shall be the maximum length of response data field supported by the mdoc, encoded as an unsigned integer. This field is further defined in 11.2. This field is mandatory for the mdoc and shall not be used by the mdoc reader.

9.2.3 Session establishment during Negotiated Handover

There can be situtations where there are security benefits for obtaining an mdoc holder permission before setting up the connection for data retrieval. When performing negotatied handover, sending the mdoc request, as defined in 10.2, at the end of the handover phase using the NFC connection, can allow the mdoc holder to confirm the connection setup and consent to the request with one action. To support this, the mdoc reader may send the SessionEstablishment message, as defined in 12.2.4, at the end of the handover phase.

The mdoc can indicate that it supports receiving the `SessionEstablishment` message during Negotiated Handover by including the `Capabilities` element in the device engagement structure.

If the mdoc indicates in the Device Engagement structure that it supports receiving the `SessionEstablishment` message during Negotiated Handover, the mdoc reader can send the `SessionEstablishment` message during Negotiated Handover.

If the mdoc does not indicate that it supports receiving the `SessionEstablishment` message during Negotiated Handover, the mdoc reader shall not send the `SessionEstablishment` message during Negotiated Handover.

Transmission of SessionEstablishment during negotiated handover shall be implemented as a TNEP service as defined in the [TNEP specification]. The Negotiated Handover service shall be closed and the service with service name [urn:placeholder] shall be established. The content shall be the SessionEstablishment message. When sending the SessionEstablishment message with this TNEP service, the SessionEstablishment message shall still be send during the data retrieval phase. These SessionEstablishment messages shall be the same.

Editor's note: to review and expand on the definition above.

If the mdoc receives the SessionEstablishment message during Negotiated Handover, the mdoc shall verify if this SessionEstablishment message matches the SessionEstablishment message received during the data retrieval phase. If they don't match, the mdoc shall terminate the connection.

If transmission of the SessionEstablishement message during negotiated handover does not succeed for any reason, the mdoc and mdoc reader shall proceed as if the SessionEstablishment message was not sent during Negiotiated Handover and continue as normal.

NOTE the intention of this last requirement is to ensure that sending the SessionEstablishment message during Negotiated Handover as a mechanism does not result in a failed transactions that would otherwise succeed.

9.3 Device engagement using QR code

If QR code is used for device engagement, the device engagement structure shall be transmitted as a barcode compliant with ISO/IEC 18004. The barcode is shown by the mdoc and read by the mdoc reader. The QR code shall contain a URI with "mdoc:" as scheme and the DeviceEngagement structure specified in 9.1 encoded using base64url-without-padding, according to RFC 4648, as path.

NOTE The requirements above result in the content of the QR code as "mdoc:" followed by the base64url-without-padding encoded device engagement structure.

An mdoc reader shall select one of the transmission technologies from the ones provided in the device engagement structure.

Table 2 defines the values that shall be used for the DeviceRetrievalMethod structure. Any other DeviceRetrievalMethod type is RFU.

Table 2 — DeviceRetrievalMethod parameters

	NFC	BLE	Wi-Fi Aware
type	1	2	3
version	1	1	1
options	NfcOptions	BleOptions	WifiOptions

The WifiOptions, BleOptions and NfcOptions structures shall be formatted as follows. Other RetrievalOptions are RFU.

```
WifiOptions = {
    ? 0: tstr,      ; Pass-phrase Info Pass-phrase
    ? 1: uint,      ; Channel Info Operating Class
    ? 2: uint,      ; Channel Info Channel Number
    3: bstr       ; Band Info Supported Bands
}
```

```
BleOptions = {
    0 : bool,       ; Indicates support for mdoc peripheral server mode
    1 : bool,       ; Indicates support for mdoc central client mode
    ? 10 : bstr,    ; UUID for mdoc peripheral server mode
    ? 11 : bstr,    ; UUID for mdoc client central mode
    ? 20 : bstr,    ; mdoc BLE Device Address for mdoc peripheral modes
    ? 21 : uint     ; mdoc BLE L2CAP PSM for mdoc peripheral l2cap mode
}

NfcOptions = {
    0 : uint,       ; Maximum length of command data field
    1 : uint        ; Maximum length of response data field
}
```

The contents of these fields are further defined in 11.1.2, 11.2 and 11.3.2.

9.4 Device engagement time-out

If the mdoc reader implements a time-out for the time between the transaction initialization and receiving device engagement data, the time-out should be no less than 30 seconds. The mdoc verifier may terminate the session at any time.

If the mdoc or mdoc reader implements a time-out for the time between receiving or sending device engagement data and sending or receiving the session establishment message (see 12.2.4), the time-out should be no less than 30 seconds. The mdoc holder and mdoc verifier may terminate the session at any time.

10 Data retrieval messages

10.1 General

The mdoc request and mdoc response messages for device retrieval are defined in 10.2 and 10.3. When used with the data transmission technologies defined in this document, they are encrypted and subsequently included in a session establishment or session data message, see 12.2. These session establishment and session data messages are then transferred using one of the transmission technologies specified in 11.1,11.2 and 11.3.

10.2 Mdoc request

10.2.1 Overview

In the first edition of this document no behavior was defined for how an mdoc reader and mdoc should interpret a request with multiple data elements or multiple documents present in the request. The current version of this document defines additional structures in `DeviceRequestInfo` and `DocRequestsInfo` that should be used by the mdoc reader to convey the exact ways in which its request can be satisfied.

Documents are requested using the device retrieval mdoc request structure, called `DeviceRequest`. At the core of this this request is the `DocRequest`. The `DocRequest` is a request for a single document. It contains all the required data elements that should be satisfied by the mdoc to satisfy the `DocRequest`. The `DocRequest` can contain extensions to the `DocRequestInfo` structure to provide additional information about the `DocRequest`. Features include the option to indicate that alternative data elements also satisfy the request and the possibility to indicate that the document has to be issued by one out of a set of issing authorities.

The DeviceRequest can contain multiple DocRequest. The DeviceRequestInfo can be used to provide information on how to interpret a request that contains multiple DocRequest. This is done by using the UseCase structure. Use cases represent higher-level groupings of DocRequest, outlining the different combinations of DocRequest that need to be fulfilled to satisfy a particular requirement. Satisfying a use case can have different ways in which it could be achieved, through different combinations of documents or data elements, but they all satisfy the use case. Multiple use case structures can be used in scenarios where returning different documents or data elements can result in different behavior by the mdoc verifier.

The mechanism of use case grouping can help the mdoc in determining how to communicate to the mdoc holder when different options to respond to the request exist. Since the mechanism has been designed such that is unambiguous how a use case can be satisfied, in a typical situation there is no necessity for the mdoc holders' input on how to respond to a certain use case request, but only whether to respond to it.

It is recommended for an mdoc to send a DeviceResponse structure that satisfies the DeviceRequestInfo and DocRequestInfo structures defined in this clause. If the mdoc cannot send a response that can satisfy the request from the mdoc reader, it is recommended to not send any document in the DeviceResponse.

This document does not contain requirement for user interfaces for the mdoc and mdoc reader in dealing with DeviceRequest and DeviceResponse.

10.2.2 Structure

The following is an informative diagram of the DeviceRequest, it does not include the RFU extensions points.

```
DeviceRequest
├──── version : Version
├──── docRequests : [DocRequest]+
│    └──── DocRequest
│        ├──── itemsRequest : #6.24(bstr .cbor ItemsRequest)
│        │    └──── ItemsRequest
│        │        ├──── docType : DocType
│        │        ├──── nameSpaces : NameSpaces
│        │        │    └──── NameSpace => DataElements+
│        │        │        └──── DataElementIdentifier => IntentToRetain (bool)
│        │        └──── ? requestInfo : DocRequestInfo
│        │            ├──── ? alternativeDataElements : [AlternativeDataElementsSet]+
│        │            │    └──── AlternativeDataElementsSet
│        │            │        ├──── requestedElement : [NameSpace, DataElementIdentifier]
│        │            │        └──── alternativeElementSets : [[NameSpace, DataElementIdentifier]+]+
│        │            ├──── ? issuerIdentifiers : [bstr]+
│        │            ├──── ? uniqueDocSetRequired : bool
│        │            ├──── ? maximumResponseSize : uint
│        │            ├──── ? zkRequest : ZkRequest
│        │            │    ├──── ZkRequired : bool
│        │            │    └──── systemSpecs : [ZkSystemSpec]+
│        │            │        └──── ZkSystemSpec
│        │            │            ├──── zkSystemId : tstr
│        │            │            ├──── system : tstr
│        │            │            └──── params : { * => Ext }
│        │            └──── ? tstr => Ext (for DocRequestInfo)
│        └──── ? readerAuth : ReaderAuth
├──── ? deviceRequestInfo : #6.24(bstr .cbor DeviceRequestInfo)
│    └──── DeviceRequestInfo
│       ├─── ? useCases : [UseCase]+
│       │    └─── UseCase
│       │         ├─── mandatory : bool
│       │         ├─── documentSets : [[DocRequestID]+]+
│       │         └─── ? purposeHints : PurposeHints
│       │              └─── PurposeControllerId => PurposeCode (int)
│       └─── tstr => Ext (for DeviceRequestInfo)
└─── ? readerAuthAll : [ReaderAuthAll]+
```

The device retrieval mdoc request structure shall be CBOR encoded and formatted as follows. The full integral structure is also specified in Annex H.

```
DeviceRequest = {
    "version" : Version,
    "docRequests" : [+ DocRequest],
    ? "deviceRequestInfo" : DeviceRequestInfoBytes,
    ? "readerAuthAll" : [+ReaderAuthAll],
    * tstr => RFU
}

DeviceRequestInfoBytes = #6.24(bstr .cbor DeviceRequestInfo)
```

`Version` is the version for the `DeviceRequest` structure: in the current version of this document its value shall be "1.0" if `deviceRequestInfo` and `readerAuthAll` is not present and "1.1" if one of them present. If other versions are specified in the future, the major version (see 8.3) of a `DeviceRequest` structure sent by the mdoc reader shall not be higher than the major version of the device engagement structure (see 9.1) communicated by the mdoc in the same transaction. This requirement therefore does not apply when the transaction does not use a device engagement structure. For example, there are flows in ISO/IEC 18013-7 that do not use the device engagement structure.

10.2.3 DocRequest

`docRequests` contains an array of all requested documents. Each requested document is defined using the DocRequest structure defined as:

```
DocRequest = {
    "itemsRequest" : ItemsRequestBytes,
    ? "readerAuth" : ReaderAuth,
    * tstr => RFU
}
```

`ReaderAuth` is further defined in 12.5.

`ItemRequestBytes` contains the `ItemsRequest` structure as a tagged CBOR bytestring. They have the following definition:

```
ItemsRequestBytes = #6.24(bstr .cbor ItemsRequest)

ItemsRequest = {
    "docType" : DocType,
    "nameSpaces" : NameSpaces,
    ? "requestInfo" : DocRequestInfo,
    * tstr => RFU
}

NameSpaces = {
    + NameSpace => DataElements
}

DataElements = {
    + DataElementIdentifier => IntentToRetain
}

IntentToRetain = bool
```

`docType` is the requested document type.

`NameSpaces` contains the requested data elements and the namespace they belong to.

`DataElements` contains the requested data elements identified by their data element identifier. For each requested data element, the `IntentToRetain` variable indicates whether the mdoc verifier intends to retain the received data element. The mdoc verifier shall not retain any data, including digests and signatures, or derived data received from the mdoc, except for data elements for which the `IntentToRetain` flag was set to true in the request. To retain is defined as “to store for a period longer than necessary to conduct the transaction in realtime”.

`requestInfo` element in `ItemsRequest` may be used by the mdoc reader to provide additional information on a single requested document and is defined in the following clause.

10.2.4 Additional document request info

DocRequestInfo is defined as:

```
DocRequestInfo = {
    ? "alternativeDataElements": [+ AlternativeDataElementsSet]
    ? "issuerIdentifiers": [+IssuerIdentifier],
    ? "uniqueDocSetRequired": UniqueDocSetRequired,
    ? "maximumResponseSize": MaximumResponseSize,
    ? "zkRequest": ZkRequest,
    * tstr => Ext
}

AlternativeDataElementsSet = {
    "requestedElement": ElementReference,
    "alternativeElementSets": [+AlternativeElementSet],
    * tstr => RFU
}

AlternativeElementSet = [+ ElementReference]
ElementReference = [NameSpace, DataElementIdentifier]

IssuerIdentifier = bstr
UniqueDocSetRequired = bool
MaximumResponseSize = uint
```

Note For readability purposes the `RequestInfo` structure in the 2021 edition of this document has been renamed to `DeviceRequestInfo` to distinguish from the new `DocRequestInfo` structure. This is a purely editiorial change to the CDDL since the identifier in the `ItemsRequest` structure for the `DocRequestInfo` structure has been kept unchanged as “requestInfo”.

If present, `DocRequestInfo` shall contain at least one key-value pair. The following key-value pairs are defined in this document. The `DocRequestInfo` structure can also be used for application-specific extensions.

AlternativeDataElements

The AlternativeDataElements structure in the DocRequestInfo structure can be used to indicate that data elements in the NameSpaces structure can have alternative data elements that also be used to fulfill the DocRequest.

The alternativeDataElements structure is an array of AlternativeDataElementsSet. Each AlternativeDataElementsSet defines alternative data elements for one of the data elements in the NameSpaces structure.

An AlternativeDataElementsSet is a map with to elements. requestedElement identifies the data element that alternatives are defined for. The data element is identifier using the ElementReference, which is an array that contains the NameSpace and the DataElementIdentifier. The requestedElement shall also be present in the NameSpaces structure.

alternativeElementSets contains an array of AlternativeElementSet. Each AlternativeElementSet is considered by the mdoc verifier to be an alternative to the requestedElement, with an AlternativeElementSet consisting of one or more data elements.

This means that instead of the requestedElement, all of the data elements of one of the AlternativeElementSet has to be returned to meet the requirement from the mdoc verifier.

The order of the data elements in AlternativeElementSets indicates the order of preference, with the first element in the array indicating the highest preference. The mdoc reader should give the most privacy-preserving alternative the highest priority.

The intent_to_retain value of the requestedElement also applies to the alternative data elements.

Note Since a single DeviceResponse structure does not allow data elements with the same data element identifier and namespace to be returned, when a single data element is used to satisfy multiple AlternativeDataElements structures, that data element is still only returned once in the response structure.

An example of the use of this structure is:

EXAMPLE Either age_over_18 or age_in_years or birth_date is sufficient to satisfy the DocRequest, with the order of preference being age_over_18 then age_in_years then birth_date.

```
DocRequestInfo =
{
    "alternativeDataElements" : [
        {
            "requestedElement": [ "org.iso.18013.5.1", "age_over_18"],
            "alternativeElementSets": [
                [
                    [ "org.iso.18013.5.1", "age_in_years"],
                ],
                [
                    [ "org.iso.18013.5.1", "birth_date"],
                ],
            ]
        }
    ]
}
```

IssuerIdentifiers

The `IssuerIdentifiers` structure in the `DocRequestInfo` structure can be used to indicate which issuers the mdoc reader will accept for this `DocRequest`. If the `IssuerIdentifiers` structure is present, an mdoc that does not meet the requirement for the `IssuerIdentifiers` structure does not satisfy the mdoc reader’s request.

The `IssuerIdentifiers` structure is an array of `IssuerIdentifier`. The format of an `IssuerIdentifier` is the value of `KeyIdentifier` element of the `AuthorityKeyIdentifier` element of an X.509 certificate encoded as byte string.

An mdoc only satisfies the `DocRequest` if one of the `IssuerIdentifier` matches the `KeyIdentifier` of the `AuthorityKeyIdentifier` in one of the certificates in in the `x5chain` element in the header of the `IssuerAuth` header of the mdoc.

Note If the mdoc reader includes the subject key identifier of the certificate that it uses to verify an mdoc issued by a particular Issuer, it will match the authority key identifier of one of the certificates in the `x5chain` of the `IssuerAuth` header.

UniqueDocSetRequired

If `UniqueDocSetRequired` is set to True, this is an indication by the mdoc reader that it expects a single response to this exact `DocRequest`

If `UniqueDocSetRequired` is set to False, this is an indication by the mdoc reader that it can receive multiple instances of responses to this exact `DocRequest`. This can support use cases where it can be useful to return multiple documents of the same type. Examples include vehicle licences for trailer and car, multiple diplomas and multiple concert tickets.

If `UniqueDocSetRequired` is not present, it is undefined whether the mdoc reader supports processing multiple responses to a single `DocRequest`.

Presence of the `UniqueDocSetRequired` element to indicate True or False is recommended for the mdoc reader.

NOTE 1 ISO/IEC 18013-5:2021 supports responding with multiple mdocs for a single DocRequest, but does not define whether the mdoc reader supports processing multiple responses to a single DocRequest.

MaximumResponseSize

The `MaximumResponseSize` element can be used by the mdoc reader to indicate the maximum response size it can accept.

The value of `MaximumResponseSize` is the value in bytes of the unencrypted `DeviceResponse`.

The mdoc is recommended to not send a `DeviceResponse` that exceeds the value of `MaximumResponseSize`. The behavior of the mdoc reader when it receives a `DeviceResponse` that exceeds the size of `MaximumResponseSize` is out of scope of this document.

zkRequest

`zkRequest` can be used to indicate support for Zero Knowledge Proof (ZKP), see 10.2.7 for the details.

10.2.5 Additional device request info

`DeviceRequestInfo` may be used by the mdoc reader to provide additional information on the set of documents requested in the mdoc request. This document specifies one key-value pair as defined below. The `DeviceRequestInfo` structure can also be used for application-specific extensions. If present, `DeviceRequestInfo` shall contain at least one key-value pair.

```
DeviceRequestInfo = {
    ? "useCases": [+UseCase],
    * tstr => Ext
}

UseCase = {
    "mandatory" : bool,
    "documentSets" : [+ DocumentSet],
    ? "purposeHints" : PurposeHints,
    * tstr => RFU
}

PurposeHints = {
    + PurposeControllerId => PurposeCode
}

PurposeControllerId = tstr
PurposeCode = int

DocumentSet = [+ DocRequestID]

DocRequestID = uint
```

UseCases

The `UseCases` structure in the `DeviceRequestInfo` structure can be used to indicate how it expects an mdoc to respond to a request containing multiple DocRequests. It can also be used to provide a purposehint for each of these requests.

The `UseCases` structure is an array that contain one or more `UseCase`. To satisfy the request from the mdoc reader, the mdoc:

- has to satisfy at least one `UseCase`
- has to satisify each `UseCase` where the `mandatory` element in the `UseCase` structure is set to `True`.
- may satisfy each `UseCase` where the `mandatory` element in the `UseCase` structure is set to `False`.

The `documentsSet` element in the `UseCase` structure is used to indicate which `DocRequest` must be satisfied to satisfy the `UseCase`. `documentSets` is an array that contains `DocumentSet`. Exactly one `DocumentSet` has to be satisfied to satisfy the `documentsSet`.

`DocumentSet` is an array that contains one or more `DocRequestID`. The `DocRequestID` references the `DocRequests` in the `docRequests` array, identified by its position starting from 0. All `DocRequest` in a `DocumentSet` have to be satisfied to satisfy the request from the mdoc reader.

A `DocRequest` is considered satisfied if all the data elements in the `NameSpaces` structure of that `DocRequest` are returned, except when `AlternativeDataElements` or `IssuerIdentifiers` change that requirement

Each `UseCase` optionally contains `PurposeHints`. The goal of `PurposeHints` is to help the user to distinguish the different UseCase in the request, but other context either as part or outside of the transaction may be necessary to exactly understand what the use case is.

`PurposeHints` is a map that contains one or more `PurposeCode`. Each `PurposeCode` has a `PurposeControllerId` that identifies the entity that defines the meaning of the `PurposeCode`. The `PurposeControllerId` should use a reverse domain name. `PurposeHints` is a map to support scenarios where an mdoc reader does not know exactly which `PurposeControllerId` the mdoc can parse. The design assumes that the mdoc parses only a single `PurposeCode` if multiple `PurposeCode` are provided.

It is recommended for the mdoc reader to include `PurposeHints`. If `PurposeHints` is not included, the mdoc holder may not have the information necessary to determine which use cases to satisfy.

This document defines 3 purposes where the `PurposeControllerId` is "org.iso.jtc1.sc17". The purposes are:

Identity verification, `PurposeCode` = 1

Driving privilege verification, `PurposeCode` = 2

Age verification, `PurposeCode` = 3

Note work is ongoing to define additional structures and extensions to provide context for the used to decide whether it should satisfy a use case.

10.2.6 Mdoc reader authentication

`ReaderAuthAll` and `ReaderAuth` are used for mdoc reader authentication as defined in 12.5.

If the device engagement phase uses a `DeviceEngagement` structure (which is the case for all flows defined in this document) and the mdoc did not set `ReaderAuthAllSupport` to true in the `DeviceEngagement` structure, `ReaderAuthAll` should not be present.

10.2.7 ZKP

An mdoc reader can indicate support for a Zero Knowledge Proof (ZKP) mechanism using the `zkRequest` `DocRequestInfo` parameter. `ZkRequest` is defined as:

```
ZkRequest = {
    "systemSpecs": [+ ZkSystemSpec],
    "zkRequired": ZkRequired,
    * tstr => RFU
}

ZkRequired = bool

ZkSystemSpec = {
    "zkSystemId": ZkSystemId
    "system": ZkSystem,
    "params": ZkParams,
    * tstr => RFU
}

ZkSystem = tstr
ZkSystemId = tstr
ZkParams = { * tstr => Ext}
```

Each `ZkSystemSpec` element defines a ZKP response type that is supported by the mdoc verifier. `ZkSystem` is a tstr that defines the ZKP type, `ZkParams` contains the parameter used by that system. Its contents are out of scope of this specification. `ZkSystemId` is an identifier set by the mdoc reader that is used in the response to refer to a specific `ZkSystemSpec` as defined in the request.

When the mdoc verifier indicates that it supports a ZKP, they mdoc may return the document using a ZKP using the ZKP mechanisms defined for the DeviceResponse.

ZkRequired is used by the mdoc reader to indicate that only a ZKP response satisisfies the request from the mdoc reader.

For compatibility reasons it's recommended to only use arrays, maps, tstr, int and bool for the values in ZkParams. This is equivalent to the following CDDL:

```
ZkParams = limited_map

limited_map = {* tstr => limited_value}
limited_value = tstr / bool / int / limited_array / limited_map
limited_array = [* limited_value]
```

Editor's note: Examples to be added

10.2.8 Request processing logic

This document does not require processing logic for the additional request elements. However, the following logic is an example of how an mdoc can determine which data elements to return to satisfies the request from the mdoc reader.

EXAMPLE processing logic, this example only considers whether a response satisfies the requested documents and elements, it does not consider additional logic that the mdoc may use to determine its response, for example consent, reader authentication, reader authorization etc.

1. For each DocRequest, determine if it can be satisfied. A DocRequest is satisfied if:
    a. the mdoc issuing authority is included in the IssuerIdentifiers structure (if present)
    b. the Doctype matches the requested DocType
    c. if ZkRequest is present and ZkRequired is set to true, the mdoc supports one of the ZkSystemSpec in the request.
    d. for each data element in NameSpaces one of the following is true:
        i. The element itself is returned.
        ii. The element is a requestedElement in an AlternativeDataElementsSet and all of the data elements in one of the AlternativeElementSet in that AlternativeDataElementsSet is returned.
2. For each DocumentSet in the UseCase, determine if all the DocRequests corresponding to the DocRequestID elements in the DocumentSet can be satisfied. Otherwise, the DocumentSet cannot be satisfied.
3. For each UseCase, determine if at least one of the DocumentSet in the UseCase can be satisfied, otherwise the UseCase cannot be satisfied. If a UseCase is mandatory and it cannot be satisfied, the DeviceRequest cannot be satisfied.
4. If there are one or more UseCases that can be satisfied: return all UseCases indicated as mandatory and optionally return a UseCase that is not mandatory. For each satisfied UseCase, return exactly one DocumentSet and for that DocumentSet return all the documents. For each of those documents, return all the data elements necessary to satisfy the corresponding DocRequest (see step 1).

An example of a device retrieval mdoc request can be found in D.4.1.1.

10.3 Mdoc response

10.3.1 Overview

Documents are returned using the device retrieval mdoc response structure, called DeviceResponse. The DeviceResponse can contain one or more documents.

Each returned document only contains the data elements that the mdoc decides to return, using the selective disclosure mechanism used by the mdoc data model. The returned document also contains the MSO and the mdoc authentication structure to provide the security features in the mdoc response.

10.3.2 Structure

The following is an informative diagram of the `DeviceResponse`, it does not include the RFU extension points.

```
DeviceResponse
├─── version : tstr
├─── ? documents : [Document]+
│    └─── Document
│         ├─── docType : DocType
│         ├─── issuerSigned
│         │    ├─── ? nameSpaces : NameSpace => [IssuerSignedItemBytes]+
│         │    │    └─── IssuerSignedItemBytes = #6.24(bstr .cbor IssuerSignedItem)
│         │    │         └─── IssuerSignedItem
│         │    │              ├─── digestID : uint
│         │    │              ├─── random : bstr
│         │    │              ├─── elementIdentifier : DataElementIdentifier
│         │    │              └─── elementValue : DataElementValue
│         │    └─── issuerAuth : IssuerAuth
│         ├─── deviceSigned
│         │    ├─── nameSpaces : DeviceNameSpacesBytes
│         │    │    └─── DeviceNameSpacesBytes = #6.24(bstr .cbor DeviceNameSpaces)
│         │    │         └─── DeviceNameSpaces
│         │    │              └─── NameSpace => DeviceSignedItems
│         │    │                   └─── DeviceSignedItems
│         │    │                        └─── DataElementIdentifier => DataElementValue
│         │    └─── deviceAuth
│         │         ├─── deviceSignature : DeviceSignature
│         │         └─── OR deviceMac : DeviceMac
│         └─── ? errors
│              └─── NameSpace => ErrorItems
│                   └─── ErrorItems
│                        └─── DataElementIdentifier => ErrorCode (int)
├─── ? zkDocuments : [ZkDocument]+
│    └─── ZkDocument
│         ├─── documentData : ZkDocumentDataBytes
│         │    └─── ZkDocumentDataBytes = #6.24(bstr .cbor ZkDocumentData)
│         │         └─── ZkDocumentData
│         │              ├─── docType : DocType
│         │              ├─── zkSystemId: ZkSystemId
│         │              ├─── timestamp : full-date
│         │              ├─── ? issuerSigned : NameSpace => [ZkSignedItem]+
|       |           |    └─── ZkSignedItem
|       |           |         ├─── elementIdentifier : DataElementIdentifier
|       |           |         └─── elementValue : DataElementValue
|       |           ├─── ? deviceSigned : NameSpace => [ZkSignedItem]+
|       |           |    └─── ZkSignedItem
|       |           |         ├─── elementIdentifier : DataElementIdentifier
|       |           |         └─── elementValue : DataElementValue
|       |           └─── ? msoX5chain : COSE_X509
|       └─── proof : bstr
├─── ? documentErrors : [DocumentError]+
|    └─── DocumentError
|         └─── DocType => ErrorCode (int)
└─── status : uint
```

The device retrieval mdoc response structure shall be CBOR encoded and formatted as follows. The full integral structure is also specified in Annex H.

```
DeviceResponse = {
    "version" : tstr,
    ? "documents" : [+Document],
    ? "zkDocuments" : [+ ZkDocument],
    ? "documentErrors": [+DocumentError],
    "status" : uint,
    * tstr => RFU
}
```

`version` is the version for the `DeviceResponse` structure. In the current version of this document its value shall be “1.0”. If other versions are specified in the future, the major version (see

8.3) of a `DeviceResponse` structure shall not be higher than the major version of the device engagement structure (see 9.1) communicated by the mdoc in the same transaction. The major version of a `DeviceResponse` structure shall also not be higher than the major version of the mdoc request (see 10.2) to which it is a response.

`documentErrors` is defined in 10.3.6.

`status` is defined in 10.3.5.

`zkDocuments` is defined in 10.3.4.

`documents` contains an array of all returned `Document` as defined in the next clause:

10.3.3 Document

```
Document = {
    "docType" : DocType,
    "issuerSigned" : IssuerSigned,
    "deviceSigned" : DeviceSigned,
    ? "errors" : Errors,
    * tstr => RFU
}
```

In the `Document` structure, the document type of the returned document is indicated by the `docType` element. The document type shall match the document type as indicated in the issuer data authentication (see 12.3) and mdoc authentication structures (see 12.4).

`errors` can contain error codes for data elements that are not returned and is further defined in 10.3.6.

`IssuerSigned` contains the mobile security object for issuer data authentication and the data elements protected by issuer data authentication:

```
IssuerSigned = {
    "issuerAuth" : IssuerAuth,
    ? "nameSpaces" : IssuerNameSpaces,
    * tstr => RFU
}

IssuerNameSpaces = {
    + NameSpace => [ + IssuerSignedItemBytes ]
}

IssuerSignedItemBytes = #6.24(bstr .cbor IssuerSignedItem)

IssuerSignedItem = {
    "digestID" : uint,
    "random" : bstr,
    "elementIdentifier" : DataElementIdentifier,
    "elementValue" : DataElementValue
    * tstr => RFU
}
```

The `IssuerAuth` structure is defined in 12.3.4.

`nameSpaces` contains the returned data elements as part of their corresponding namespaces.

Individual data elements are returned as `IssuerSignedItem`. The `digestID` and `random` are defined in 12.3.5. `elementIdentifier` is the data element identifier and `elementValue` the data element value. Each `IssuerSignedItem` is returned as part of the corresponding namespace in the `IssuerNameSpaces` structure.

`DeviceSigned` contains the mdoc authentication structure and the data elements protected by mdoc authentication:

```
DeviceSigned = {
    "nameSpaces" : DeviceNameSpacesBytes,
    "deviceAuth" : DeviceAuth,
    * tstr => RFU
}

DeviceNameSpacesBytes = #6.24(bstr .cbor DeviceNameSpaces)

DeviceNameSpaces = {
    * NameSpace => DeviceSignedItems
}

DeviceSignedItems = {
    + DataElementIdentifier => DataElementValue
}

DeviceAuth = {
    "deviceSignature" : DeviceSignature //     ; "//" means OR
    "deviceMac" : DeviceMac
    * tstr => RFU
}
```

nameSpaces contains the returned data elements as part of their corresponding namespaces. nameSpaces is a mandatory element because the element is authenticated using mdoc authentication. The DeviceNameSpaces structure can be an empty structure. The DeviceAuth structure contains either the DeviceSignature or the DeviceMac element, both are defined in 12.4.

The mdoc shall not include two or more data elements, either IssuerSignedItem or DeviceSignedItem, with the same DataElementIdentifier in a single NameSpace and Document.

DeviceSignedItems contains the data element identifiers and values. DeviceSignedItems is returned as part of the corresponding namespace in DeviceNameSpaces.

An example of a device retrieval mdoc response can be found in D.4.1.2.

An example of using DeviceSigned to sign data by an mdoc can be found in D.7.

10.3.4 ZKP

An mdoc may return a document using a ZKP mechanism by returning it in ZkDocument instead of Document. A document may only be returned using a ZKP mechanism if the following conditions are true:

- In the ZkRequest DocRequestInfo parameter in the Docrequest, the mdoc verifier indicates that it supports a ZKP with the ZkSystem parameter used in the ZkDocument.
- The mdoc is capable of generating the ZKP

The ZkDocument structure is defined as:

```
ZkDocument = {
    "documentData": ZkDocumentDataBytes,
    "proof": bstr,
    * tstr => RFU
}

ZkDocumentDataBytes = #6.24(bstr .cbor ZkDocumentData)

ZkDocumentData = {
    "docType": DocType,
    "zkSystemId": ZkSystemId,
    "timestamp": full-date,
    ? "issuerSigned" : ZkNameSpaces,
    ? "deviceSigned" : ZkNameSpaces,
    ? "msoX5chain": COSE_X509,
    * tstr => RFU
}

ZkNameSpaces = {
    + NameSpace => [ + ZkSignedItem ]
}

ZkSignedItem = {
    "elementIdentifier": DataElementIdentifier,
    "elementValue": DataElementValue,
    * tstr => RFU

}
```

documentData contains the data for which the ZKP proof is returned, proof contains the proof itself. The content of proof is out of scope of this document.

ZkSystemId is used to indicate which ZKP system is used, by referencing one of the ZkSystemSpec from the request.

Editor's note: todo to what extent to define the values for ZkDocumentData

10.3.5 mdoc response status

The `status` element shall contain one of the status codes in Table 3. If the mdoc returns a status code different than 0, it shall not return any documents.

Table 3 — Response status

Status code	Status message	Description	Actions required
0	OK	Normal processing. This status message shall be returned if no other status is returned	No specific action required
10	General error	The mdoc returns an error without any given reason.	The mdoc reader may inspect the problem. The mdoc reader may continue the transaction.
11	CBOR decoding error	The mdoc indicates an error during CBOR decoding that the data received is not valid CBOR. Returning this status code is optional.	The mdoc reader may inspect the problem. The mdoc reader may continue the transaction.
12	CBOR validation error	The mdoc indicates an error during CBOR validation, e.g. wrong CBOR structures. Returning this status code is optional.	The mdoc reader may inspect the problem. The mdoc reader may continue the transaction.

10.3.6 mdoc response error codes

If the device retrieval mdoc response structure does not include some data element or document requested in the device retrieval mdoc request, an mdoc may return an error code as part of the `DocumentErrors` or `Errors` structures.

`DocumentErrors` can contain error codes for documents that are not returned. The structure has the following definition:

```
DocumentError = {
    DocType => ErrorCode
}

ErrorCode = int
```

`Errors` can contain error codes for documents that are not returned. The structure has the following definition:

```
Errors = {
    + NameSpace => ErrorItems
}

ErrorItems = {
    + DataElementIdentifier => ErrorCode
}
```

If present, the `ErrorCode` element shall contain an error code from Table 4. An `ErrorCode` is specific to the document or data element requested within a namespace.

Table 4 — Data handling error

Error code	Error code message	Description
0	Data not returned	The mdoc does not provide the requested document or data element without any given reason. This element may be used in all cases.
Other positive integers	See description	RFU
Negative integers	See description	These error codes may be used for application-specific purposes.

11 Data retrieval transmission technologies

11.1 Data retrieval using Bluetooth® low energy (BLE)

11.1.1 General

Bluetooth®[1] low energy (BLE), implemented according to the Bluetooth Core Specification[10], may be used for device retrieval. An mdoc shall support at least version 4.0 of this specification. An mdoc should at least support version 4.2 and LE Data Packet Length Extension. An mdoc reader shall support version 4.2 and LE Data Packet Length Extension. An mdoc reader is recommended to at least support version 5.0 and LE 2M PHY.

Using BLE as a transmission technology consists of two phases, connection setup and data retrieval. During connection setup, the mdoc and mdoc reader connect to each other. After the connection is set up, data retrieval can be initiated.

This specification defines 4 different modes for transmission using BLE:

Using GATT for transmission:

— mdoc central client mode, where the mdoc supports the Central role and acts as GATT client.

— mdoc peripheral server mode, where the mdoc supports the Peripheral role and acts as GATT server.

Using L2CAP for transmission:

— mdoc central l2cap mode, where the mdoc supports the Central role and uses L2CAP directly.

— mdoc peripheral l2cap mode, where the mdoc supports the Peripheral role and uses L2CAP directly.

During the device engagement phase, an mdoc, and optionally the mdoc reader, indicates whether it supports the Central role, the Peripheral role or both, and which of the modes they support.

NOTE It is possible for the mdoc or mdoc reader to support both the Central and Peripheral role, until data retrieval has started.

[1] Bluetooth is the trademark of a product supplied by the Bluetooth Special Interest Group. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO/IEC of the product named. Equivalent products may be used if they can be shown to lead to the same results.

Data transmission can use GATT, which is mandatory to be supported by the mdoc and mdoc reader, or directly use L2CAP, which is optional to be supported by the mdoc and mdoc reader. GATT and L2CAP are implemented according to the Bluetooth Core Specification[10].

BLE secure connections can be used if supported by both the mdoc and mdoc reader. However, an mdoc or an mdoc reader shall not require the use of BLE secure connections. Security of the transferred data is ensured by the security mechanisms specified in Clause 12.

All BLE session information shall be removed after each transaction.

11.1.2 Device engagement contents

This subclause describes the requirements for the contents of the device engagement information. Reference [13] gives further informative guidance.

Device engagement using NFC

For device engagement using NFC, the BLE alternative carrier configuration record Record Type Name shall be “application/vnd.bluetooth.le.oob”. The Supplement to the Bluetooth Core Specification[11] specifies data types used for OOB data blocks. The following requirements apply to the presence of the data types in the OOB data block.

— LE Role (0x1C). This type is mandatory. It shall be used by the mdoc and mdoc reader to indicate which LE roles they support or select.

— LE Device Address (0x1B). This field is recommended. If it is available, the connection process can take a shorter time compared to using the UUID to identify the correct device to connect to.

— Complete List of 128-bit Service UUIDs (0x07). The requirements for including this field are defined below. This field shall be used to transfer the UUIDs used for connection setup, if applicable.

— Sevice Data – 16-bit UUID (0x16). Should be present if the mdoc or mdoc reader supports mdoc specific BLE data as defined below and that data is non-empty. The 16-bit UUID used shall be the XXXX which is an UUID allocated by the Bluetooth SIG for ISO/IEC JTC SC17. The service data shall be the bytes of the MdocBleServiceData CBOR structure as defined below.

The following requirements apply for including the UUID field during NFC device engagement:

— for Negotiated Handover, if the mdoc reader supports mdoc central client mode, it shall include a UUID in the Handover Request message, to be used for mdoc central client mode. This is the UUID that the mdoc reader advertises.

— for Negotiated Handover, if the mdoc chooses to use mdoc peripheral server mode, it shall include a UUID in the Handover Select message, to be used for mdoc peripheral server mode. This is the UUID that the mdoc advertises.

— for Static Handover, the mdoc shall send one UUID in the handover select message, to be used for mdoc central client mode or mdoc peripheral server mode as appropriate. This is the UUID that the mdoc, mdoc reader or both advertise.

The MdocBleServiceData shall be CBOR encoded and formatted as follows:

```
MdocBleServiceData = {
    ? 0 : MdocBlePsm,
    * int => RFU
}

MdocBlePsm = int
```

The value of MdocBlePsm contains the Protocol Service Multiplexer (PSM) that the mdoc or mdoc reader exposes and that can be used by the mdoc or mdoc reader to connect to when one of the l2cap modes is used. An mdoc or mdoc reader shall only advertise this field when it supports the Peripheral Role. When the l2cap modes are supported, use of the `MdocBlePsm` field is recommended since it can speed up the connection process by not needing to first interact using the GATT layer.

Other data types may be included in the OOB data block.

NOTE When encoding the OOB data blocks, note that the Supplement to the Bluetooth Core Specification specifies that certain values should be little-endian, in Clause 1 it is stated that: "All numerical multi-byte entities and values associated with the following data types shall use little-endian byte order."

Device engagement using QR code

For device engagement using QR code, the mdoc shall indicate which modes it supports using the fields in the `BleOptions` structure as specified in 9.3. The UUIDs in the `BleOptions` structure shall be encoded using variant 1 ('10x'b) as a 16 octets element with the byte order as specified in RFC 4122, section 4.1.2. The value of the BLE Device Address field shall use the same encoding as the LE Device Address field used in device engagement using NFC as defined earlier in this subclause.

The UUID for peripheral server mode shall be present if mdoc peripheral server mode is supported and shall not be present if peripheral server mode is not supported. This is the UUID that the mdoc advertises.

The UUID for client central mode shall be present if mdoc central client mode is supported and shall not be present if central client mode is not supported. This is the UUID that the mdoc reader advertises.

The BLE Device Address field may be present if mdoc peripheral server mode is supported and it shall not be present if peripheral server mode is not supported. If it is available, the connection process can take a shorter time compared to using the UUID to identify the correct device to connect to.

The L2CAP PSM field should be present when if mdoc peripheral l2cap mode is supported by the mdoc. The L2CAP PSM field shall not be present if the mdoc does not support mdoc peripheral l2cap mode. The content shall be the Protocol Service Multiplexer (PSM) exposed by the mdoc that the mdoc reader can connect to. Use of the PSM field is recommended since it can speed up the connection process by not needing to first interact using the GATT layer.

11.1.3 GATT based modes

11.1.3.1 Connection setup

The UUIDs used shall be 16-byte UUIDs that are unique for the transaction. The Peripheral device shall broadcast the service with the UUID as received or sent during device engagement in the advertising packet. The Central device is then able to scan for the UUID and connect to the advertised service. However, the Central device may use a different mechanism to identify the Peripheral device.

NOTE 1 BLE stacks in mobile devices can use scan filter and caching methods to manage congested environments and manage scan intervals for device energy consumption control. This can influence the connection time required when using UUIDs for the identification of the Peripheral device.

NOTE 2 Finding the correct device to connect to is purely a practical problem. Connecting to the wrong mdoc reader does not have security implications, since due to the security methods described in clause 12, the mdoc and mdoc reader will not setup a session with the wrong mdoc reader. Note however, that these mechanisms do not provide complete protection against a bad actor aiming to cause a denial of service attack by advertising as a fake mdoc reader.

If the mdoc indicates during device engagement that it supports both modes, the mdoc reader should select the mdoc central client mode.

To ensure that the mdoc is connected to the correct mdoc reader, the mdoc may verify the Ident characteristic as described in 11.1.3.2. The Ident characteristic value shall be calculated using the following procedure:

Use HKDF as defined in RFC 5869 with the following parameters:

— Hash: SHA-256,

— IKM: `EdeviceKeyBytes` (see 12.2.4),

— salt: (no salt value is provided),

— info: “BLEIdent” (encoded as a UTF-8 string),

— L: 16 octets.

If the mdoc receives the Ident characteristic from the mdoc reader and it does not match the expected value, the mdoc shall terminate the connection.

NOTE 3 The purpose of the Ident characteristic is only to verify whether the mdoc is connected to the correct mdoc reader before starting data retrieval. If the mdoc is connected to the wrong mdoc reader, session establishment will fail. Connecting and disconnecting to an mdoc reader takes a relatively large amount of time and it is therefore fastest to implement methods to identify the correct mdoc reader to connect to and not to rely purely on the Ident characteristic to identify the correct mdoc reader.

After connection is setup, the GATT client may check to see if the GATT server supports the L2CAP transmission profile (see 11.1.4) and, if so, use it for data transmission.

11.1.3.2 Service definition

Table 4 shows characteristics which the mdoc service shall contain if the mdoc is the GATT server. The services may contain other characteristics and properties besides the ones required in Table 5.

NOTE Supporting the Write property next to Write Without Response property can solve some interoperability issues. Using Write Without Response has a higher transmission rate.

Table 5 — mdoc service characteristics

Characteristic name	UUID	Mandatory properties
State	00000001-A123-48CE-896B-4C76973373E6	Notify, Write Without Response
Client2Server	00000002- A123-48CE-896B-4C76973373E6	Write Without Response
Server2Client	00000003- A123-48CE-896B-4C76973373E6	Notify

Table 6 shows characteristics which the mdoc reader service shall contain if the mdoc reader is the GATT server. The services may contain other characteristics and properties besides the ones required in Table 6.

Table 6 — mdoc reader service characteristics

Characteristic name	UUID	Mandatory properties
State	00000005- A123-48CE-896B-4C76973373E6	Notify, Write Without Response
Client2Server	00000006- A123-48CE-896B-4C76973373E6	Write Without Response

Server2Client	00000007- A123-48CE-896B-4C76973373E6	Notify
Ident	00000008- A123-48CE-896B-4C76973373E6	Read

Each service characteristic having the Notify property shall contain the Client Characteristic Configuration Descriptor, with UUID '0x29 0x02' and default value of '0x00 0x00'. This value shall be set to '0x00 0x01' by the GATT client to get notified for the characteristic associated to this descriptor.

11.1.3.3 Connection state

After the connection is setup, the GATT client shall subscribe to notifications of characteristic 'State' and 'Server2Client'. For performance reasons, the GATT client should request for an MTU as high possible. After these steps, the GATT client shall make a write without response request to 'State' where it sets the value to 0x01. This tells the GATT server that the GATT client is ready for the transmission to start.

The connection state is indicated by the 'State' characteristic. It is encoded as 1-byte binary data. Table 7 describes the different connection state values, which are communicated using Write Without Response and Notify.

Table 7 — Connection state values

Command	Data	Sender	Description
Start	0x01	GATT client	This indicates that the mdoc reader may/will begin transmission.
End	0x02	mdoc, mdoc reader	Signal to finish/terminate transaction. The mdoc reader shall use this value to signal the end of data retrieval. Both the mdoc and the mdoc reader can use this value at any time to terminate the connection. See also 12.2.4 for more information and other options for session termination. Note if the mdoc uses this command for session termination, care has to be taken that the session is not terminated while transmission of a SessionData message is ongoing.

11.1.3.4 Data retrieval

Editor's note: To check what the performance impact is of supporting write with response.

Data retrieval shall start by signalling the 'Start' value to the 'State' characteristic.

The data sent shall be the `SessionEstablishment` or `SessionData` messages as defined in 12.2.4.

When the GATT client sends a message to the GATT server, it shall send this message using the Write Without Response command via the 'Client2Server' characteristic.

When the GATT server sends a message to the GATT client, it shall send this message using the Notify command via the 'Server2Client' characteristic.

When sending a message, the message shall be divided in parts for which the maximum size does not exceed any of the following:

- 3 bytes less than the negotiated MTU size
- the maximum length of the attribute value allowed by the Bluetooth Core Specification [10].

Note The maximum length of the attribute value specified by the BLE specification Bluetooth Core Specification version 5.2 is 512 bytes.

Within those limits, the size of each part is recommended to be as large possible to reduce transmission time.

The first byte of each part is either 0x01, which indicates more messages are coming, or 0x00, to indicate it is the last part of the message.

The sequence of messages shall be repeated as long as necessary to finish data retrieval.

Figure 4 shows the informative sequence diagram for the data retrieval phase as described in this subclause.

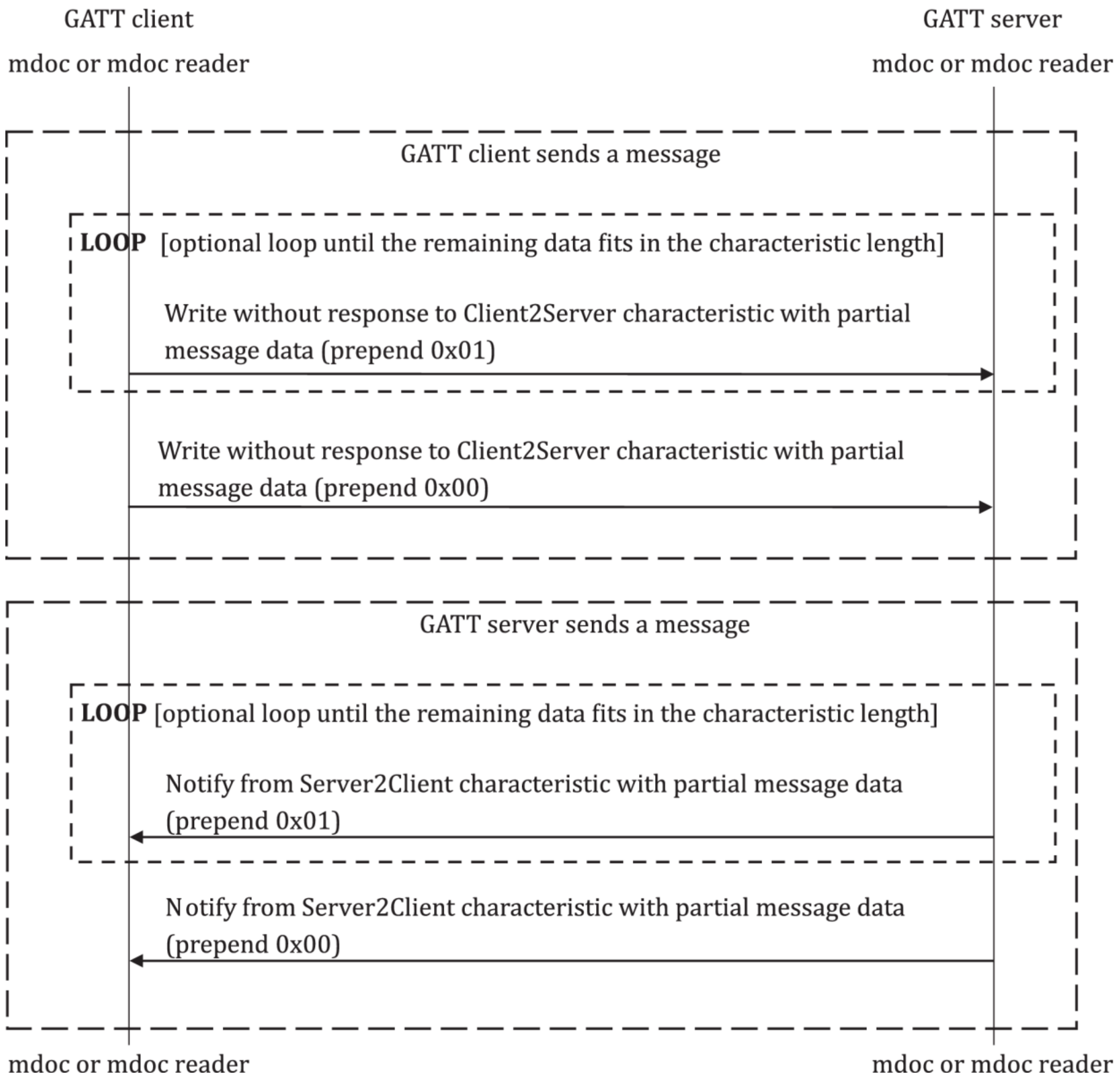
GATT client
mdoc or mdoc reader
GATT server
mdoc or mdoc reader
GATT client sends a message
LOOP [optional loop until the remaining data fits in the characteristic length]
Write without response to Client2Server characteristic with partial message data (prepend 0x01)
Write without response to Client2Server characteristic with partial message data (prepend 0x00)
GATT server sends a message
LOOP [optional loop until the remaining data fits in the characteristic length]
Notify from Server2Client characteristic with partial message data (prepend 0x01)
Notify from Server2Client characteristic with partial message data (prepend 0x00)
mdoc or mdoc reader
mdoc or mdoc reader

Figure 4 — Data transfer sequence diagram

11.1.3.5 Connection closure

After data retrieval, the GATT client shall disconnect from the GATT server.

11.1.3.6 Connection re-establishment

In case of a lost connection before the ‘State’ characteristic has been set to a value of 0x01 (e.g. the transmission has not yet started), the mdoc and mdoc reader should terminate their current BLE session and try to reconnect according to 11.1.3.1.

In case of a lost connection after the 'State' characteristic has been set to value 0x01 (e.g. the transmission of data has started), a connection shall not be re-established, and a completely new mdoc transaction shall be initiated if required.

11.1.4 l2cap based modes

The entity with the Central role can indicate support for the L2CAP transmission mode by including the PSM during device engagement, by advertising the L2CAP GATT characteristic or both.

When the PSM is shared by advertising the GATT characteristic the GATT server shall indicate support for the L2CAP transmission profile by making the L2CAP characteristic available next to the characteristics defined in 11.1.3.2. Table 7 shows the characteristic which the GATT server shall expose. The contents of the characteristics shall be the Protocol Service Multiplexer (PSM).

Table 7 — L2CAP service characteristics

Service	Characteristic name	UUID	Mandatory properties
mdoc	L2CAP	0000000A- A123-48CE-896B-4C76973373E6	Read
mdoc reader	L2CAP	0000000B- A123-48CE-896B-4C76973373E6	Read

If the information necessary to setup the L2CAP is shared during the engagement phase or as part of the GATT layer, the mdoc and mdoc reader may setup the L2CAP connection by establishing an L2CAP connection-oriented channel according to the Bluetooth Core Specification, Version 5.2, December 2019, Vol 3, Part A, section 4[10].

After the connection is setup, data transmission can start. The contents of the data sent shall be the `SessionEstablishment` or `SessionData` messages as defined in 12.2.4. The data sent shall be pre-pended with a header that indicates the length of the data (excluding the header element). This header element shall be an integer encoded as a fixed-length 4 byte element.

11.2 Data retrieval using near field communication (NFC)

NFC may be used for device retrieval. In case NFC is used, the mdoc shall support PICC mode and the mdoc reader shall support PCD mode.

An mdoc and mdoc reader shall support short-length fields as specified in ISO/IEC 7816-4:2020, 5.2 and should support extended-length fields as specified in ISO/IEC 7816-4:2020, 5.2. An mdoc shall indicate the maximum length of command data fields and of response data fields it supports during device engagement, as specified in 9.3 for device engagement using QR code and 9.2.2 for device engagement using NFC. The two fields shall indicate the maximum length of the command and response data fields as defined in ISO/IEC 7816-4:2020, 5.2 supported by the mdoc.

NOTE 1 For device engagement using QR code, this information is in the NfcOptions map. For device engagement using NFC, this information is in the NFC Handover Select message.

NOTE 2 The minimum and maximum possible values for the command data field limit are 'FF' and 'FF FF', i.e. the limit is between 255 and 65 535 bytes (inclusive). The minimum and maximum possible values for the response data limit are '01 00' and '01 00 00', i.e. the limit is between 256 and 65 536 bytes (inclusive).

The mdoc reader shall respect the data field size limitations of the mdoc.

NOTE 3 The mdoc reader determines the size of both the command and the response APDUs and can, therefore, make sure that its APDU size limitations (if any) are not being violated.

An mdoc and mdoc reader shall support command chaining and response chaining as specified in ISO/IEC 7816-4:2020, 5.3.

NOTE 4 Even if extended-length APDUs are supported by the mdoc and the mdoc reader, using command or response chaining can still be necessary, because it cannot be guaranteed that an mdoc reader request or an mdoc response fits in a single APDU.

The Application IDentifier (AID) of the mdoc shall be 'A0 00 00 02 48 04 00'.

NOTE 5 The AID of the mdoc application consists of the registered application provider identifier (RID) ('A0 00 00 02 48') followed by the proprietary application identifier extension (PIX) ('04 00').

An mdoc application shall be selected using the SELECT command defined in ISO/IEC 7816-4 with the AID listed above. Table 8 and Table 9 specify the SELECT command and response APDUs.

Table 8 — SELECT command

CLA	INS	P1-P2	Lc field	Data field	Le field
‘00’	‘A4’	’04 0C’	‘07’	‘A0 00 00 02 48 04 00’	Absent

Table 9 — SELECT response

Data field	SW1 – SW2
Absent	See ISO/IEC 7816-4:2020, Table 61

After the mdoc application is selected, the mdoc reader can start data retrieval. The mdoc reader shall use the ENVELOPE command with INS = ‘C3’, specified in ISO/IEC 7816-4, to communicate the `SessionEstablishment` en `SessionData` messages as defined in 12.2.4 to the mdoc. These messages shall be encapsulated in a data object ‘53’ as specified in ISO/IEC 7816-4:2020, 11.7.2. The mdoc shall use the ENVELOPE response to communicate the `SessionData` messages as defined in 12.2.4 to the mdoc reader. These messages shall be encapsulated in a data object ‘53’ as well.

Table 10 and Table 11 specify the ENVELOPE command and response APDUs.

Table 10 — ENVELOPE command

CLA	INS	P1-P2	Lc field	Data field	Le field
‘00’ or ‘10’, as defined in ISO/IEC 7816-4:2020, 5.4.1	‘C3’	‘0000’	Length of data field	Data object ‘53’ or data object fragment	See below

Table 11 — ENVELOPE response

Data field	SW1 – SW2
Data object ‘53’ or absent if an error occurred on ISO/IEC 7816-4:2020 protocol level	See ISO/IEC 7816-4:2020, Table 121, in particular ’61 XY’.

For oversize incoming payload (from an mdoc reader to an mdoc), several ENVELOPE commands shall be chained as specified in ISO/IEC 7816-4:2020, 5.3. For oversize outgoing payload (from an mdoc to an mdoc reader), response chaining shall be used as specified in ISO 7816-4:2020, 5.3 and further detailed below, using one or more GET RESPONSE commands and responses.

Regarding the value of Le in the ENVELOPE and GET RESPONSE commands:

— For all ENVELOPE commands in a chain except the last one, Le shall be absent, since no data is expected in the response to these commands;

— For the last ENVELOPE command in a chain, Le shall be set to the maximum length of the response data field that is supported by both the mdoc and the mdoc reader. The mdoc reader shall encode Le as specified in ISO/IEC 7816-4:2020, 5.2;

— For the last ENVELOPE command or for a GET RESPONSE command,

  — if Le ≥ the number of available bytes, the mdoc shall include all available bytes in the response and set the status words to ’90 00’;

  — if Le < the number of available bytes ≤ Le + 255, the mdoc shall include as many bytes in the response as indicated by Le and shall set the status words to ’61 XX’, where XX is the number of available bytes remaining. The mdoc reader shall respond with a GET RESPONSE command where Le is set to XX;

  — if the number of available bytes > Le + 255, the mdoc shall include as many bytes in the response as indicated by Le and shall set the status words to ’61 00’. The mdoc reader shall respond with a GET RESPONSE command where Le is set to the maximum length of the response data field that is supported by both the mdoc and the mdoc reader.

If the NFC connection is lost during data retrieval, a completely new mdoc transaction (including device engagement) shall be initiated.

11.3 Data retrieval using Wi-Fi Aware

11.3.1 General

Wi-Fi Aware may be used for device retrieval. Wi-Fi Aware shall be implemented according to the Wi-Fi Alliance Neighbor Awareness Networking Specification. The data retrieval using Wi-Fi Aware consists of three phases, connection setup, data retrieval and closure.

11.3.2 Connection setup

Wi-Fi Aware is setup using the information exchanged during device engagement. The Wi-Fi Alliance Neighbor Awareness Networking Specification, Version 3.1 describes the connection setup process. The Wi-Fi Alliance Neighbor Awareness Networking Specification, section 12 describes the connection setup process for NFC Negotiated Connection Handover and NFC Static Connection Handover. The structure used to transfer the required info if QR code is used for device engagement is described below.

The service name shall be calculated by the mdoc and mdoc reader for each transaction using the following mechanism.

Use HKDF as defined in RFC 5869 with the following parameters:

— Hash: SHA-256,

— IKM: `EDeviceKeyBytes` (see 12.2.4),

— salt: (no salt value is provided),

— info: “NANService” (encoded as a UTF-8 string),

— L: 16 octets.

The output of the HKDF shall be converted to base16 according to RFC 4648 to get the service name.

EXAMPLE “94AB45CDBDEF675162183B12AC35EFAA”.

Use of a cipher suite as defined by the Wi-Fi Alliance Neighbor Awareness Networking Specification is mandatory. The mdoc reader shall support the NCS-SK-128 and NCS-PK-2WDH-128 cipher suites, as specified by the Wi-Fi Alliance Neighbor Awareness Networking Specification. The mdoc shall support at least the NCS-SK-128 cipher suite and should support the NCS-PK-2WDH-128 cipher suite.

If NFC is used for device engagement, either the Pass-phrase Info or the DH Info shall be explicitly transferred from the mdoc to the mdoc reader during device engagement according to the Wi-Fi Alliance Neighbor Awareness Networking Specification, section 12.

NOTE 1 Since the NCS-PK-2WDH cipher suite requires both the mdoc and the mdoc reader to exchange ephemeral public keys during device engagement, it can only be used if NFC Negotiated Connection Handover is used and not if NFC Static Connection Handover or QR code is used for device engagement.

NOTE 2 The Wi-Fi Alliance Neighbor Awareness Networking Specification references RFC 8110 for the Diffie-Hellman key exchange used in the NCS-PK-2WDH cipher suites. Note that RFC 8110 mandates the support for the group nineteen curve, which is the P-256 curve. This is also the curve indicated as “256-bit random ECP group” (and having value 19) in the table referenced for the value of the D-H Key Group in Wi-Fi Alliance Neighbor Awareness Networking Specification, section 12.1.

If QR code is used for device engagement, the data used for connection setup that would be transferred as part of the Wi-Fi Aware Carrier Configuration Record if NFC were used, as defined in the Wi-Fi Alliance Neighbor Awareness Networking Specification, is transferred as part of the `WifiOptions` structure (see 9.3). Table 12 describes for each element in the `WifiOptions` structure what the corresponding field would be in the Wi-Fi Aware Carrier Configuration Record. The mdoc and mdoc reader shall comply with the requirements in Wi-Fi Alliance Neighbor Awareness Networking Specification regarding the presence and values of these fields, with the exception of the presence of the Pass-phrase field, for which the presence and associated behaviour is defined below.

Table 12 — WifiOptions elements

Key in WifiOptions	Corresponding field in Wi-Fi Carrier Configuration Record	Corresponding Sub-Field
0	Pass-phrase Info	Pass-phrase
1	Channel Info	Operating Class
2	Channel Info	Channel Number
3	Band Info	Supported Bands

Presence of the pass-phrase (i.e. the key-value pair with key = 0) in the `WifiOptions` structure is optional. When the Pass-phrase field is absent in `WifiOptions` structure, the mdoc and mdoc reader shall calculate the pass-phrase using the following mechanism.

Use HKDF as defined in RFC 5869 with the following parameters:

— Hash: SHA-256,

— IKM: `EdeviceKeyBytes` (see 12.2.4),

— salt: (no salt value is provided),

— info: “NANPassphrase” (encoded as a UTF-8 string),

— L: 32 octets.

The output of the HKDF calculation shall be converted using base64url-without-padding according to RFC 4648 to get the pass-phrase.

The pass-phrase field in the `WifiOptions` structure should only be present if the resulting pass-phrase contains at least 12 bytes of entropy.

During the Wi-Fi Aware service discovery procedure, the mdoc shall serve as the Service Publisher, and the mdoc reader shall serve as the Service Subscriber.

Once the Wi-Fi Aware service discovery is completed, the mdoc reader shall initiate the data path setup, and serve as the NDP Initiator; while the mdoc shall serve as the NDP Responder. The transport protocol and port number shall be transferred as part of the NDPE attribute according to Wi-Fi Alliance Neighbor Awareness Networking Specification*,* Version 3.1, January 2020*,* section 6.2.7. The mdoc and mdoc reader should use an ephemeral link-local IPv6 for each connection.

Since the IP address is transferred as part of the Wi-Fi Aware connection setup, using the IPv6 Neighbor Discovery Protocol is not necessary. The Neighbor Discovery Protocol should therefore be disabled to significantly improve connection time setup.

11.3.3 Data retrieval

When Wi-Fi Aware is used, mdoc data is transferred using the HTTP protocol, with the mdoc serving as the HTTP and TCP servers, and the mdoc reader serving as the HTTP and TCP client. The data retrieval shall use the HTTP POST method to transfer mdoc data. HTTP request messages shall have the following structure:

```
POST /mdoc HTTP/1.1
Host: [IPv6 address of the mdoc]
Content-Length: [content length]
Content-Type: application/cbor

[SessionEstablishment or SessionData message]
```

HTTP successful response message shall have the following structure:

```
HTTP/1.1 200 OK
Content-length: [content length]
Content-type: application/cbor

[SessionData message]
```

The `SessionEstablishment` and `SessionData` messages are defined in 12.2.4.

HTTP error responses are specified in RFC 7231, section 6.1.

Editor’s note: is something higher than 1.1 allowed?

11.3.4 Closure

An mdoc or mdoc reader shall close the connection after receiving the session termination code, see 12.2.4.

12 Security mechanisms

12.1 Overview

The security of mdoc data exchanged with an mdoc reader is designed to preserve the triad of confidentiality, integrity, and authenticity by design and by default.

The security architecture aims to achieve four distinct goals.

a) Protection against forgery: data elements are signed by the issuing authority (IA), or they are MACed or signed by the mdoc. In the first case, the degree of protection against forgery depends on the degree to which the IA's keys are protected. In the second case, the degree of protection against forgery depends on the degree to which the mdoc's secure area protects the data elements and the mdoc authentication keys. It also depends on the degree to which the IA's keys are protected, as the mdoc authentication keys are signed by the IA. Minimizing the validity period of the mdoc limits the value of the data.

b) Protection against cloning: the mdoc produces a signature or message authentication code over session data. The private key used to authenticate the session data is stored only in the mdoc. The corresponding public key in turn is signed by the IA. The degree of protection against cloning depends on the degree to which the mDL authentication key is protected.

c) Protection against eavesdropping: communications between mdoc and mdoc reader are encrypted and authenticated. Device engagement uses a separate communication channel to mitigate the risk of Man in the Middle (MITM) attacks. In addition, the mdoc reader can detect MITM attacks by validating the anti-cloning signature or message authentication code, which is created using a key for which the public part is signed by the IA in the mobile security object (MSO, see 12.3.4). If mdoc reader authentication is used, the mdoc can detect MITM attacks before returning any data.

d) Protection against unauthorized access: an mdoc is protected from unauthorized access by an mdoc reader by multiple mechanisms. The encryption key used for communications between the mdoc and mdoc reader is derived from an ephemeral key pair from both the mdoc and mdoc reader. The ephemeral public key of the mdoc is shared only through short-range device engagement. This ensures data is only transferred between the mdoc and a particular mdoc reader. The mdoc can optionally authenticate the mdoc reader by means of an mdoc reader certificate and a signature created by the mdoc reader using the corresponding private key. The mdoc reader certificate is signed by a certificate authority trusted by the mdoc for this purpose.

Table 13 and Table 14 describe the security mechanisms that can be implemented and how they achieve the goals described above.

Issuer data authentication and mdoc authentication shall be implemented. All transmission technologies for device retrieval specified in this document use session encryption. Implementation of reader authentication is optional.

When verifying any of the security mechanisms defined in this clause, an mdoc reader shall follow the procedures defined in 12.7.

The MSO includes update information and validity time frames which enable the mdoc reader to check the freshness of the data. The issuing authority shall define appropriate periods of validity that balance freshness with offline capability, considering that a shorter validity period mitigates certain security risks. The MSO can contain revocation status information.

Table 13 — Security mechanisms

Security mechanisms	Reference
Session encryption	12.2
Issuer data authentication	12.3
mdoc authentication	12.4
mdoc reader authentication	12.5

Table 14 — Security goals and mechanisms

Security goal	Functionality	Mechanism
Protection against forgery	Authenticate the origin of mDL data	Issuer data authentication
	Verify mDL data has not changed from issuing authority	Issuer data authentication
	Verify how up to date the mDL data is	Issuer data authentication
Protection against cloning	Protect against cloning of mDL/ binding mDL data to a specific device	mdoc authentication
Protection against eavesdropping	Preserve confidentiality of mDL data	Session encryption
	Prevent unnoticed alteration of communication	Session encryption mdoc authentication
Protection against unauthorized access	Prevent unauthorized access of mDL data	Close-range device engagement with session encryption
	Prevent unauthorized access of mDL data	mdoc reader authentication[b]
[b] This is an optional mechanism.		

12.2 Session encryption

12.2.1 Purpose

Encrypting with authentication of the mdoc requests and mdoc responses with the session key protects mdoc data from eavesdropping and alteration.

12.2.2 Applicability

This mechanism is applicable for an mdoc using device retrieval with one of the data retrieval mechanisms defined in this document.

12.2.3 Description

Session encryption uses standard ephemeral key ECDH to establish session keys for authenticated symmetric encryption.

12.2.4 Procedure

The following steps shall be performed as part of session encryption.

1. Device engagement. The mdoc generates a new ephemeral key pair (EDeviceKey.Priv, EDeviceKey.Pub), and includes the cipher suite identifier, the identifier of the elliptic curve to be used for key agreement and the EDeviceKey public point, as part of the device engagement structure as defined in 9.1.

2. Session establishment. The mdoc reader generates a new ephemeral key pair (EReaderKey.Priv, EReaderKey.Pub) using the elliptic curve identified by the mdoc. Session keys are derived independently by the mdoc and the mdoc reader as specified in 12.2.5.

The mdoc reader encrypts the mdoc request with the appropriate session key and sends it to the mdoc together with EReaderKey.Pub in a session establishment message.

The mdoc uses the data from the session establishment message to derive the session keys and decrypts the mdoc request.

3. Session data. The mdoc encrypts the mdoc response with the appropriate session key and sends it to the mdoc reader in a session data message.

   The mdoc reader and mdoc optionally exchange further session data messages containing additional mdoc requests and mdoc responses. If so, these requests and responses are encrypted by the mdoc reader and the mdoc using their respective session keys.

4. Session termination. The mdoc and mdoc reader can initiate session termination.

   An mdoc or mdoc reader can initiate session termination in one of the following ways:

   - By sending the status code for session termination in a session data message, this can be sent together with an mdoc request or response.
   - If BLE GATT is used for data transmission, the session termination can also be initiated by sending the 'End' command defined in 11.1.3.3.

   When an mdoc reader receives a session data message, it shall respond with either a status code or another request.

   An mdoc can send the mdoc response without a session termination code. This allows for receiving further requests or waiting to see if the mdoc reader doesn't respond with codes 10 or 11, thereby verifying that no errors occurred in receiving the response.

   When a session is terminated, the mdoc and mdoc reader shall perform at least the following actions:

   — destruction of session keys and related ephemeral key material;

   — closure of the communication channel used for data retrieval.

   The mdoc or mdoc reader may terminate after a time-out of no activity of receiving or sending session establishment or session data messages. This document does not mandate implementation of such a time-out, but if it is implemented, the time-out for no activity implemented by the mdoc and mdoc reader should be no less than 300 s.

The session establishment message shall be CBOR encoded and formatted as follows:

```
SessionEstablishment = {
     "eReaderKey" : EReaderKeyBytes,
     "data" : bstr                   ; Encrypted mdoc request
    * tstr => RFU
}
```

The session data messages shall be CBOR encoded and formatted as follows:

```
SessionData = {
    ? "data" : bstr          ; Encrypted mdoc response or mdoc request
    ? "status" : uint        ; Status code
    * tstr => RFU
}
```

The contents of the `data` element in the session establishment and session data messages are defined in 12.2.5.

The mdoc and mdoc reader ephemeral keys shall be encoded as COSE_Key as defined in RFC 9052; further requirements are defined in 12.6.2. The structures that contain the mdoc and mdoc reader ephemeral keys shall be CBOR encoded and formatted as follows:

```
EDeviceKey = COSE_Key                                ; Containing
EDeviceKey.Pub
EReaderKey = COSE_Key                                ; Containing
EReaderKey.Pub
EDeviceKeyBytes = #6.24(bstr .cbor EDeviceKey)
EReaderKeyBytes = #6.24(bstr .cbor EReaderKey)
```

When cipher suite 1 (see 12.6.2) is used, one of the curves from Table 17 shall be used in `EDeviceKey` and `EReaderKey`. Only curves with the purpose ECDH shall be used.

The possible values of the status code are defined in Table 15. If status code 10 or 11 is returned, the `data` element shall not be present in that session data message. An mdoc reader should respond with status code 10 or 11 when applicable.

Table 15 — SessionData status codes

Status code	Description	Action required
10	Error: session encryption	The session shall be terminated.
11	Error: CBOR decoding	The session shall be terminated.
20	Session termination	The session shall be terminated.

An example for session encryption can be found in D.5.1.

12.2.5 Cryptographic operations

The following operations shall be performed if cipher suite 1 (see 12.6.2) is used.

To calculate the session keys, the mdoc and the mdoc reader shall perform ECKA-DH (Elliptic Curve Key Agreement Algorithm – Diffie-Hellman) as defined in BSI TR-03111. The inputs shall be the EDeviceKey.Priv and EReaderKey.Pub for the mdoc and EReaderKey.Priv and EDeviceKey.Pub for the mdoc reader. The Z_{AB} output defined in BSI TR-03111 shall be used to derive two keys, SKReader and SKDevice.

SKReader shall be derived using HKDF as defined in RFC 5869 with the following parameters:

— Hash: SHA-256,

— IKM: Z_{AB},

— salt: SHA-256(`SessionTranscriptBytes`),

— info: “SKReader” (encoded as a UTF-8 string),

— L: 32 octets.

SKDevice shall be derived using HKDF as defined in RFC 5869 with the following parameters:

— Hash: SHA-256,

— IKM: Z_{AB},

— salt: SHA-256(`SessionTranscriptBytes`),

— info: "SKDevice" (encoded as a UTF-8 string),

— L: 32 octets.

`SessionTranscriptBytes` is defined in 12.6.1.

For encryption AES-256-GCM (GCM: Galois Counter Mode) as defined in NIST SP 800-38D shall be used. The mdoc reader shall encrypt its mdoc requests with SKReader, the mdoc shall encrypt its mdoc responses with SKDevice. Therefore, both the mdoc and the mdoc reader need to generate both session keys in order to be able to decrypt the messages they send and also decrypt the messages they receive.

The IV (Initialization Vector defined in NIST SP 800-38D) used for encryption shall have the default length of 12 bytes for GCM, as specified in NIST SP 800-38D. The IV shall be the concatenation of the identifier and the message counter (identifier || message counter). The identifier shall be an 8-byte value. The mdoc reader shall use the following identifier: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00. The mdoc shall use the following identifier: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01. The mdoc and mdoc reader shall keep a separate message counter for each session key. The message counter value shall be a 4-byte big-endian unsigned integer. For the first encryption with a session key, the message counter shall be set to 1. Before each following encryption with the same key, the message counter value shall be increased by 1. A message counter value shall never be reused in any future encryption using the same key. The AAD (Additional Authenticated Data defined in NIST SP 800-38D) used as input for the GCM function shall be an empty string. The plaintext used as input for the GCM function shall be mdoc request or mdoc response. The value of the `data` element in the session establishment and session data messages as defined in 12.2.4 shall be the concatenation of the ciphertext and all 16 bytes of the authentication tag (ciphertext || authentication tag).

12.3 Issuer data authentication

12.3.1 Purpose

The purpose of issuer data authentication is to confirm that the mdoc data is issued by the issuing authority and that it has not changed since issuance.

12.3.2 Applicability

This mechanism is applicable for an mdoc supporting device retrieval.

12.3.3 Description

Issuer data authentication is implemented by way of a digital signature over mdoc data, calculated by the issuing authority infrastructure using a public-private (asymmetric) key pair.

The isuing authority infrastructure calculates a message digest for each data element present on the mdoc and includes all digests in the mobile security object (MSO), defined in 12.3.4. The issuing authority infrastructure then digitally signs the MSO using a private key that is kept secret by and adds the digital signature to the mdoc data.

The public key belonging to the private key used for the digital signature is provided as part of a certificate. When the mdoc is presented to an mdoc reader, the mdoc reader retrieves this certificate. The mdoc reader shall then perform the inspection procedure as described in 12.7.1.

12.3.4 Signing method and structure for MSO

An mdoc digital signature is generated over the mobile security object (MSO). The MSO shall be CBOR encoded and formatted as follows:

```
IssuerAuth = COSE_Sign1     ; The payload is MobileSecurityObjectBytes

MobileSecurityObjectBytes = #6.24(bstr .cbor MobileSecurityObject)

MobileSecurityObject = {
    "version" : tstr,
    "digestAlgorithm" : tstr,
    "valueDigests" : ValueDigests,
    "deviceKeyInfo" : DeviceKeyInfo,
    "docType" : tstr,
    "validityInfo" : ValidityInfo
    ? "status": Status
    * tstr => RFU
}

DeviceKeyInfo = {
    "deviceKey" : DeviceKey
    ? "keyAuthorizations" : KeyAuthorizations,
    ? "keyInfo" : KeyInfo
    * tstr => RFU
}

DeviceKey = COSE_Key

KeyAuthorizations = {
    ? "nameSpaces" : AuthorizedNameSpaces
    ? "dataElements" : AuthorizedDataElements
    * tstr => RFU
}

AuthorizedNameSpaces = [+ NameSpace]
AuthorizedDataElements = {+ NameSpace => DataElementsArray}
DataElementsArray = [+ DataElementIdentifier]

KeyInfo = {
    * uint => RFU,
    * nint => Ext
}

ValueDigests = {
    + NameSpace => DigestIDs
}

DigestIDs = {
    + DigestID => Digest
}

ValidityInfo = {
    "signed" : tdate,
    "validFrom" : tdate,
    "validUntil" : tdate,
    ? "expectedUpdate" : tdate
    * tstr => RFU
}

Status = {
    ? "identifier_list" : IdentifierListInfo,
    ? "status_list : StatusListInfo,
    * tstr => RFU
}

DigestID = uint
Digest = bstr
```

The version for the `MobileSecurityObject` structure shall be "1.0" in the current version of this document. The major version (see 8.3) shall not be higher than the major version of the mdoc response (see 10.3).

The `digestAlgorithm` and `valueDigests` are the digest algorithm identifier and the digests of the data elements as further specified in 12.2.5.

Even if the issuing authority does not issue any `IssuerSignedItems` to the mDL, `the valueDigests` and the `digestAlgorithm` shall be present in the MSO. In this case `digestAlgorithm` shall be set to an arbitrary digestAlgorithm identifier according to Table 16 and `valueDigests` shall contain single a digest of matching length. It is mandatory for the issuing authority to use a value that does not allow identification, linking or tracking of the mdoc holder. For example, this can be achieved by using a random value of sufficient entropy.

`deviceKeyInfo` contains the mdoc authentication public key and information related to this key. `deviceKey` contains the public part of the key pair used for mdoc authentication (see 12.4). The `deviceKey` element is encoded as an untagged `COSE_Key` element as specified in RFC 9052; further requirements are defined in 12.6.2.

As specified in 12.4.4, an mdoc can use a DeviceKey to calculate a signature or MAC over data elements as part of mdoc authentication. Within `DeviceKeyInfo`, `KeyAuthorizations` shall contain all the elements the key may sign or MAC. Authorizations can be given for a full namespace or per data element. If authorization is given for a full namespace (by including the namespace in the `AuthorizedNameSpaces` array), that namespace shall not be included in the `AuthorizedDataElements` map. If the `KeyAuthorizations` map is present, it shall not be empty.

`KeyInfo` may contain extra info about the key. Positive integers for `KeyInfo` labels are RFU. If application-specific extensions are present, they shall use negative integers for the labels.

`DigestID` is an unsigned integer that is used to match the hashes in the MSO to the data elements in the mdoc response. The Digest ID shall be unique within a namespace. To prevent the MSO leaking information on what data elements are present on a specific mdoc, there should be no correlation between the Digest ID's used for the same data element in the same namespace in different MSO's. The value shall be smaller than 2^31.

`DocType` is the document type of the document and shall be identical to the `DocType` element in the mdoc response as defined in 10.3.

The `ValidityInfo` structure contains information related to the validity of the MSO and its signature. The `signed` element is the timestamp at which the MSO signature was created. The `validFrom` element contains the timestamp before which the MSO is not yet valid. The timestamp of `validFrom` shall be equal or later than the `signed` element.

NOTE 1 A `validFrom` element with a future date can be used for when a change of mdoc data is expected in the future, for example, a change in age data elements.

The validUntil element contains the timestamp after which the MSO is no longer valid. The value of the validUntil element shall be later than the validFrom element. The value of the validUntil element shall be equal or earlier than the value of the notAfter element in the leaf certificate in the x5chain element of the IssuerAuth structure. An mdoc reader may reject an mdoc if there is a time in the future where the leaf certificate is expired but the MSO is not.

The optional expectedUpdate element contains the timestamp at which the issuing authority infrastructure expects to re-sign the MSO (and potentially update data elements).

The timestamps in the ValidityInfo structure shall not use fractions of seconds and shall use a UTC offset of 00:00, as indicated by the character “Z”.

NOTE 2 The validUntil element determines the validity period of the MSO and therefore, the mdoc cannot be validated after this date. mdoc data elements can provide further information on the administrative validity of the mdoc. For example, if the mdoc has an expiry date data element, this date can be later than the ‘validUntil’ date of the MSO.

Since the timestamps in the ValidityInfo structure can provide linkability clues, the issuing authority infrastructure should set these timestamps with a precision that limits the linkability information. This can be done, for example, by setting the hh, mm and ss information to the same value on each provisioned mdoc.

Status is an optional structure that may be used to provide MSO revocation information to the mdoc reader, which can be compared to the information in an externally hosted MSO revocation list. Usage of the Status structure, if present, is optional for an mdoc reader. Details are defined in 12.3.6.

The MSO is encapsulated and signed by the untagged COSE_Sign1 structure as defined in RFC 9052 and identified as IssuerAuth for use in the mdoc response as defined in 10.3. Within the COSE_Sign1 structure, the payload shall be MobileSecurityObjectBytes. The external_aad field used in the Sig_structure shall be a bytestring of size zero.

The alg element (RFC 9052) shall be included as an element in the protected header.

The issuing authority infrastructure shall use one of the following signature algorithms as defined in RFC 9053 for calculating the signature over the MSO: “ES256” (ECDSA with SHA-256), “ES384” (ECDSA with SHA-384), “ES512” (ECDSA with SHA-512) or “EdDSA” (EdDSA). ”ES256” shall be used with curves P-256 and brainpoolP256r1. “ES384” shall be used with curves P-384, brainpoolP320r1 and brainpoolP384r1. “ES512” shall be used with curves P-521 and brainpoolP512r1. “EdDSA” shall be used with curves Ed25519 and Ed448. For verifying the signature, the mdoc reader shall support all of these signature algorithms and curves.

The recommendation in RFC 9053 on the use of deterministic ECDSA signatures does not apply to this document.

The certificate containing the public key belonging to the private key used to sign the MSO shall be included as an x5chain element as described in RFC 9360. It shall be included as an unprotected header element. The x5chain element shall include at least one certificate and may contain more.

An mdoc reader should support the presence of the x5chain element in the protected header. An mdoc should include the x5t element as defined in RFC 9360 in the protected header. If present, the x5t element shall use the SHA-256 algorithm.

NOTE 3 The identifier for the x5chain element can be found in the IANA registry for COSE Header Parameters.

NOTE 4 While there are no security concerns with the use of the unprotected header for the x5chain element in the context of this specification, there is a general tendency to put these certificates in the protected header instead of the unprotected header. To ensure backwards compatibility with the first edition of this specification, this requirement is not changed. However in the future this requirement could change, which is why the recommendation was added to also support the x5chain element in the protected header.

An example can be found in D.5.2.

12.3.5 Message digest function

The issuing authority infrastructure shall use one of the following digest algorithms: SHA-256, SHA-384 or SHA-512 as specified in ISO/IEC 10118-3. In the `digestAlgorithm` key-value pair in the MSO, the algorithms used shall be identified as defined in Table 16.

Table 16 — Digest algorithm identifiers

Digest algorithm	digestAlgorithm identifier
SHA-256	"SHA-256"
SHA-384	"SHA-384"
SHA-512	"SHA-512"

A digest shall be calculated separately for each data element present on the mdoc and stored in the MSO. The same digest algorithm shall be used for all data elements. Digests are identified by the combination of the `NameSpace` (see 8.1) and the `DigestID` (see 12.3.4). The input for the digest calculation shall be the `IssuerSignedItemBytes` element (see 10.3.3). Each `IssuerSignedItem` shall also contain an unpredictable random or pseudorandom value. This value shall be different for each `IssuerSignedItem` and shall have a minimum length of 16 bytes. The purpose of this value is to ensure that the digest value of the `IssuerSignedItem` by itself does not provide any information about its contents.

NOTE It is not necessary for the mdoc reader to retrieve all the data present on the mdoc to verify the received mdoc data; it can verify the signature over the whole MSO and use the Digest ID and namespace for each received data element to find and verify the digest for those data elements in the MSO.

12.3.6 MSO revocation

12.3.6.1 Introduction

An MSO may contain the `Status` structure, which contains MSO revocation information. The `Status` structure contains a reference to an MSO revocation list. The MSO revocation list is a COSE_Sign1 structure that indicates whether a particular MSO is revoked or not. The `Status` structure contains all the information necessary for the mdoc reader to determine whether the MSO revocation list is authentic.

Two mechanisms can be used to encode the revocation information: as an identifier list or as a status list. The identifier list mechanism revokes an MSO based on whether the issuer-defined identifier in the MSO is present on the identifier list. The status list mechanism revokes an MSO based on whether the bit of the issuer-defined bit position in the MSO is set to true in the status list.

The status list mechanism is specified in the Token Status List (draft-ietf-oauth-status-list-12) specification. This document also defines the identifier list mechanism, largely based on the requirements from the Token Status List specification such that there is a large commonality between the status list and identifier list mechanism.

When used for an mdoc with the mDL doctype, no other status besides 'revoked' shall be used. For a status list this implies that only values "VALID" and "INVALID", defined in the Token Status List specification are used. For the identifier list, it implies that only revoked MSOs (as opposed to for example temporarily suspended MSOs) are put on the identifier list.

Once an MSO is revoked, the MSO shall be permanently revoked and as long as the MSO is still considered valid according to the `validUntil` element, no MSO revocation list shall be issued that indicates a different status than revoked.

Verification of the MSO revocation list is optional for the mdoc reader, but if done, verification shall follow the verification requirements specified in the Token Status List specification, on top of the requirements status in this document.

12.3.6.2 Status structure in the MSO

To enable an mdoc reader to verify the MSO revocation status, the MSO shall contain a `Status` element. When implementing the identifier list mechanism, the `status` element shall contain the `identifier_list` element as defined below. When implementing the status list mechanism, the `status` element shall contain the `status_list` element as defined below.

The `identifier_list` and `status_list` in the MSO may contain the `Certificate` element. If the `Certificate` element is present, it shall contain a certificate containing the public key that signed the top-level certificate in the `x5chain` element in the MSO revocation list structure. The mdoc reader shall use that certificate as trust point for verification of the `x5chain` element in the MSO revocation list structure. If the `Certificate` element is not present, the top-level certificate in the `x5chain` element shall be signed by the certificate used to sign the certificate in the x5chain element of the MSO. In the context of an mDL, that is the IACA certificate.

Editor’s note: The presence of the Certificate element is under consideration for removal if as part of the DIS ballot no comments are made in its favor.

12.3.6.3 MSO revocation list

An MSO revocation list shall be implemented according to the Token Status List specification as a Status List Token in CWT format, since the `IssuerAuth` structure is a CWT. The following requirements apply to the MSO revocation list for both the identifier list and status list mechanism:

- The `exp` claim shall be present.
- The `ttl` claim may be present.
- The `aggregation_uri` claim in the IdentifierList or StatusList claim may be present.
- The CWT shall be a COSE_Sign1 object using one of the following signature algorithms for calculating the signature: “ES256” (ECDSA with SHA-256), “ES384” (ECDSA with SHA-384), “ES512” (ECDSA with SHA-512) or “EdDSA” (EdDSA). ”ES256” shall be used with curves P-256 and brainpoolP256r1. “ES384” shall be used with curves P-384, brainpoolP320r1 and brainpoolP384r1. “ES512” shall be used with curves P-521 and brainpoolP512r1. “EdDSA” shall be used with curves Ed25519 and Ed448.
- The CWT shall contain the `x5chain` in the protected header that contains the certificate or chain of certificates used to verify the signature of the MSO revocation list.
- The extended key usage OID specified in the Token Status List specification may be used for both the status list as well as the identifier list signing certificate. It is recommended for mdoc readers to support the extended key usage OID specified in the Token Status List specification and it is not recommended for issuer authority infrastructure to make the extended key usage field critical when using the extended key usage OID specified in the Token Status List specification.

The modc issuing authority can use the aggregation_uri claim to indicate support for the aggregation mechanism as defined in the Token Status List specification.

12.3.6.4 Identifier list details

In deviation to the requirements of the Token Status List specification, the following requirements apply to the identifier list mechanism:

- The value of the `type` claim shall be “application/identifierlist+cwt”
- The `StatusList` claim shall not be present in the CWT claims set
- The `IdentifierList` structure defined below shall be present as a claim in the CWT claims set using the key 65530

The `IdentifierList` structure is a CBOR structure with the following CDDL:

```
IdentifierList = {
    "identifiers" : { * Identifier => IdentifierInfo },
    ? "aggregation_uri" : Aggregation_uri
    * tstr => RFU
}

IdentifierInfo = { tstr/int => RFU }
Identifier = bstr
Aggregation_uri = tstr
```

Presence of the `Identifier` in the `IdentifierList` indicates that the MSO that contains that identifier in the status element is revoked.

The `Aggregation_uri` claim is defined in section 9.2 of the Token Status List specification

The identifier list content-type shall be “application/identifierlist+cwt” further following the requirements as defined in section 8.2 of Token Status List specification

The following requirements apply to the identifier_list element in the MSO:

The `identifier_list` element is a CBOR structure with the following CDDL:

```
IdentifierListInfo = {
    "id" : Identifier,
    "uri": URI,
    ? "certificate": Certificate
    * tstr => RFU
}

URI = tstr
Certificate = bstr
```

To prevent the `Identifier` from being used as a correlation across presentations, it is recommended to be unique per MSO, this could be achieved by using a random identifier per MSO.

The value of the `Identifier` field shall be unique per MSO, other details for the `Identifier` are out of scope for this specification.

Annex D.6 contains examples of the identifier list.

12.3.6.5 Status list details

The following requirements apply to the status list:

The `bits` element in the `StatusList` structure shall be set to 1.

The following requirements apply to the status_list element in the MSO:

The `status_list` element shall follow the requirements for the `StatusListInfo` structure as defined in the token status list specification with the addition of the optional `Certificate` element defined in 12.3.6.2.

To prevent the status index from being as a correlation across presentations, it is recommended that the combination of status index and URI is unique per MSO.

Annex D.6 contains examples of the status list.

12.4 mdoc authentication

12.4.1 Purpose

The security objective of mdoc authentication is to prevent cloning of the mdoc and to mitigate man in the middle attacks.

12.4.2 Applicability

This mechanism is applicable for an mdoc using device retrieval.

12.4.3 Description

The mdoc private key, which belongs to the mdoc public key stored in the MSO, is used to authenticate the mdoc. It is also used to authenticate the response data contained in the `DeviceSignedItems` structure (see 10.3.3). The mdoc public key is stored in the MSO, see 12.3.4. The mdoc reader assumes that the mdoc is authentic only if the authentication signature or MAC is correct.

Security requirements regarding storage of credential information, including the mdoc private key are out of scope for this document. Additional information on storage of credential information can be found in E.5.

NOTE Two mechanisms exist for mdoc authentication, MAC and ECDSA/EdDSA. MAC provides better privacy to the mdoc holder because it does not require the mdoc to produce a potentially non-repudiable signature over mdoc reader-provided data. The mdoc can always deny the MAC value to a third party because the mdoc reader could have produced it by itself. However, it is possible that the possibility to calculate a MAC is not available in all security environments on the mdoc.

12.4.4 Mechanism

The mdoc authentication key pair consists of a public and a private key (SDeviceKey.Priv, SDeviceKey.Pub). The public key is accessible through the `DeviceKey` element in the MSO. When cipher suite 1 (see 12.6.2) is used, one of the curves from Table 17 shall be used for the device key.

The mdoc authentication key shall be used to authenticate the mdoc in one of two ways: ECDH-agreed MAC or ECDSA / EdDSA signature. A single mdoc authentication key shall not be used to produce both MACs and signatures during its lifetime. An mdoc reader shall support both approaches.

The data that the mdoc authenticates is the `DeviceAuthenticationBytes` structure as defined below. The mdoc shall generate this structure and calculate either the MAC or signature. In order to verify the data, the mdoc reader shall generate the structure as well and validate the MAC or signature.

NOTE The `DeviceAuthenticationBytes` structure itself is not transferred as part of the mdoc response, only the resulting MAC or signature.

The device authentication structure shall be CBOR encoded and formatted as follows:

```
; For DeviceMac and DeviceSignature, use a null value for the payload.
; The detached content is DeviceAuthenticationBytes
DeviceMac = COSE_Mac0
```

```
DeviceSignature = COSE_Sign1

DeviceAuthenticationBytes = #6.24(bstr .cbor DeviceAuthentication)

DeviceAuthentication = [
    "DeviceAuthentication",
    SessionTranscript,
    DocType,                    ; Same as in mdoc response
    DeviceNameSpacesBytes       ; Same as in mdoc response
]
```

The `SessionTranscript` element is defined in 12.6.1.

The `DocType` and `DeviceNameSpacesBytes` shall contain the same data as in the same `Document` element in the mdoc response structure (see 10.3.3).

An mdoc shall only authenticate response data elements in `DeviceNameSpaces` if the key it is using for mdoc authentication is authorized to authenticate these elements in the `KeyAuthorizations` structure in the MSO (see 12.3.4). The mdoc reader shall validate this authorization as part of validating the mdoc authentication.

If data elements are present in `DeviceNameSpaces`, an mdoc reader shall verify whether the `KeyAuthorizations` structure contains the proper authorization.

`DeviceMac` is defined in 12.4.5.

`DeviceSignature` is defined in 12.4.6.

An example can be found in D.5.3.

12.4.5 mdoc MAC Authentication

To authenticate the mdoc with mdoc MAC authentication, the mdoc computes the MAC of the device authentication data with an ephemeral MAC key (EMacKey) derived from the mdoc authentication private key and the mdoc reader ephemeral public key.

The following operations shall be performed when cipher suite 1 (see 12.6.2) is used.

To calculate the ephemeral MAC key, the mdoc and the mdoc reader shall perform ECKA-DH (Elliptic Curve Key Agreement Algorithm – Diffie-Hellman) as defined in BSI TR-03111. The inputs shall be the SDeviceKey.Priv and EReaderKey.Pub for the mdoc and EReaderKey.Priv and SDeviceKey.Pub for the mdoc reader. The Z_{AB} output defined in BSI TR-03111 shall be used to derive the ephemeral MAC key.

EMacKey shall be derived using HKDF as defined in RFC 5869 with the following parameters:

— Hash: SHA-256,

— IKM: Z_{AB},

— salt: SHA-256(`SessionTranscriptBytes`),

— info: “EMacKey” (encoded as a UTF-8 string),

— L: 32 octets.

The MAC value is contained in the tag element within `DeviceAuth` in an untagged `COSE_Mac0` structure as defined in RFC 9052 and identified as `DeviceMac`. Within the `COSE_Mac0` structure, the payload shall have a null value. The detached content is `DeviceAuthenticationBytes`. The ‘external_aad’ field shall be a bytestring of size zero.

The `alg` element (RFC 9052) shall be included as an element in the protected header. Other elements should not be present in the protected header.

RFC 9053 describes the algorithm identifiers that shall be used in the `alg` element. "HMAC 256/256" (HMAC with SHA-256) shall be used.

NOTE mdoc MAC authentication can only be used when the mdoc authentication key and the mdoc reader ephemeral key use the same curve. Since the mdoc reader ephemeral key uses the same curve as the mdoc ephemeral key, mdoc MAC authentication can only be used for with mdocs where the mdoc authentication key matches the mdoc ephemeral key.

12.4.6 mdoc ECDSA / EdDSA Authentication

To authenticate the mdoc with mdoc ECDSA/EdDSA authentication, the mdoc signs the device authentication data with the mdoc authentication private key.

When cipher suite 1 is used (see 12.6.2) the following operations shall be performed and the mdoc shall use of the ECDSA or EdDSA curves from Table 17 for the mdoc authentication key.

The signature is contained in the signature element in an untagged COSE_Sign1 structure as defined in RFC 819052 and identified as `DeviceSignature`. Within the COSE_Sign1 structure, the payload shall have a null value. The detached content is `DeviceAuthenticationBytes`. The 'external_aad' fields shall be a bytestring of size zero.

The `alg` element (RFC 9052) shall be included as an element in the protected header. Other elements should not be present in the protected header. An mdoc shall use one of the following signature algorithms as defined in RFC 9053: "ES256" (ECDSA with SHA-256), "ES384" (ECDSA with SHA-384), "ES512" (ECDSA with SHA-512) or "EdDSA" (EdDSA). "ES256" shall be used with curves P-256 and brainpoolP256r1. "ES384" shall be used with curves P-384, brainpoolP320r1 and brainpoolP384r1. "ES512" shall be used with curves P-521 and brainpoolP512r1. "EdDSA" shall be used with curves Ed25519 and Ed448.

The recommendation in RFC 9053 on the use of deterministic ECDSA signatures does not apply to this document.

12.5 mdoc reader authentication

12.5.1 Purpose

mdoc reader authentication uses information stored in the mdoc reader to confirm that the mdoc reader and the mdoc request are authenticated.

12.5.2 Applicability

This mechanism is applicable for an mdoc reader using device retrieval.

12.5.3 Description

A private key stored in the mdoc reader is used to authenticate the mdoc reader and to authenticate the mdoc request. The mdoc reader public key is stored in a certificate which is sent to the mdoc within the mdoc request message.

12.5.4 Mechanism

The mdoc reader authentication key pair consists of a public and a private key. The public key is accessible through a certificate provided with the mdoc request.

The mdoc reader authentication key may be used to authenticate the mdoc reader by ECDSA/EdDSA signature.

How the mdoc uses the mdoc reader information after performing the mdoc reader authentication procedure is out of scope of this document.

When cipher suite 1 is used (see 12.6.2) the following operations shall be performed and the mdoc reader shall use of the ECDSA or EdDSA curves from Table 17 for the mdoc reader authentication key.

The data that the mdoc reader authenticates is the `ReaderAuthentication` or `ReaderAuthenticationAll` structure as defined below. The mdoc reader shall generate this structure and calculate the signature. To verify the data, the mdoc shall generate the structure as well and validate the signature.

The signature is contained in an untagged COSE_Sign1 structure as defined in RFC 9052 and identified as `ReaderAuth` or `ReaderAuthAll`. Within the COSE_Sign1 structure, the payload shall have a null value. The detached content is `ReaderAuthenticationBytes`. The 'external_aad' fields shall be a bytestring of size zero.

The `alg` element (RFC 9052) shall be included as an element in the protected header. An mdoc reader should use one of the following signature algorithms as defined in RFC 9053: "ES256" (ECDSA with SHA-256), "ES384" (ECDSA with SHA-384), "ES512" (ECDSA with SHA-512) or "EdDSA" (EdDSA). "ES256" shall be used with curves P-256 and brainpoolP256r1. "ES384" shall be used with curves P-384, brainpoolP320r1 and brainpoolP384r1. "ES512" shall be used with curves P-521 and brainpoolP512r1. "EdDSA" shall be used with curves Ed25519 and Ed448.

The recommendation in RFC 9053 on the use of deterministic ECDSA signatures does not apply to this document.

NOTE 1 The ReaderAuthentication structure itself is not transferred as part of the mdoc request, only the resulting signature.

An mdoc request can contain multiple `DocRequest` elements. Mdoc reader authentication can be used for each element individually or all at the same time.

The ReaderAuth structure can be used to authenticate an individual DocRequest, with the authenticated data being ReaderAuthenticationBytes. The ReaderAuthAll structure can be used to authenticate the full request, with the authenticated data being ReaderAuthenticationAllBytes.

It is recommended to use ReaderAuthAll when possible, even when authenticating a single DocRequest.

The reader authentication structure shall be CBOR encoded and formatted as follows:

```
; For ReaderAuth, use a null value for the payload.
; The detached content is ReaderAuthenticationBytes
ReaderAuth = COSE_Sign1

ReaderAuthenticationBytes = #6.24(bstr .cbor ReaderAuthentication)

ReaderAuthentication = [
    "ReaderAuthentication",
    SessionTranscript,
    ItemsRequestBytes          ; Same as in mdoc request
]

; For ReaderAuthAll, use a null value for the payload.
; The detached content is ReaderAuthenticationBytes
ReaderAuthAll = COSE_Sign1

ReaderAuthenticationAllBytes = #6.24(bstr .cbor ReaderAuthenticationAll)

ReaderAuthenticationAll = [
    "ReaderAuthenticationAll",
    SessionTranscript,
    ItemsRequestBytesAll,           ; Order as in the mdoc request
    DeviceRequestInfoBytes / null
]

ItemsRequestBytesAll = [*ItemsRequestBytes]
```

The SessionTranscript element is defined in 12.6.1.

The ItemsRequestBytes shall have the same value as the corresponding element in the mdoc request structure (see 10.2).

ItemsRequestBytesAll shall contain all ItemsRequestBytes in the DeviceRequest, ordered in the same way as the DeviceRequest.

DeviceRequestInfoBytes shall be the same as the DeviceRequestInfoBytes structure from the DeviceRequest. If it's not present, the value shall be null.

When an X.509 certificate is used to authenticate the mdoc reader public key, that certificate containing the mdoc reader public key shall be included as a x5chain element as described in RFC 9360. It shall be included as an unprotected header element. The x5chain element shall include at least one certificate and may contain more.

It is recommended that an mdoc reader supports the presence of the x5chain element in the protected header. It is recommended that an mdoc includes the x5t element as defined in RFC 9360 in the protected header. If present, the x5t element shall use the SHA-256 algorithm.

NOTE 2 The identifier for the x5chain element can be found in the IANA registry for COSE Header Parameters.

12.6 Session transcript and cipher suite

12.6.1 Session transcript

The session transcript structure is used in multiple security mechanisms for device retrieval. The session transcript shall be CBOR encoded and formatted as follows:

```
SessionTranscript = [
    DeviceEngagementBytes / null,
    EReaderKeyBytes / null,
    Handover
]

DeviceEngagementBytes = #6.24(bstr .cbor DeviceEngagement)

Handover = QRHandover / NFCHandover / Ext

QRHandover= null

NFCHandover = [
    bstr        ; Binary value of the Handover Select Message
    bstr / null ; Binary value of the Handover Request Message,
                ; shall be null if NFC Static Handover was used
]
```

```
SessionTranscriptBytes = #6.24(bstr .cbor SessionTranscript)
```

`DeviceEngagement` is defined in 9.1.

`EReaderKeyBytes` is defined in 12.2.4.

When the DeviceEngagement structure is not used in the transaction, the value of DeviceEngagementBytes is replaced with null in the session transcript. When the EReaderKey structure is not used in the transaction, EReaderKeyBytes is replaced with null in the session transcript. These situations only occur when an engagement or transmission mechanism is used different than the ones specified in this document.

NOTE This document uses both `SessionTranscript` and `SessionTranscriptBytes` in cryptographic structures.

The content of `Handover` depends on the device engagement method that was used. If device engagement using QR code (see 9.3) was used, the contents shall be `QRHandover`. If device engagement using NFC (see 9.2) was used, the contents shall be `NFCHandover`.

The first element in the `NFCHandover` array shall be the binary value of the Handover Select Message as retrieved by the mdoc reader from the mdoc. The second element in the `NFCHandover` array shall be the Handover Request Message sent by the mdoc reader to the mdoc during NFC Negotiated Handover, or null if the Handover Request Message was not present because NFC Static Handover was used.

12.6.2 Cipher suite

Device retrieval security mechanisms support multiple cipher suites to indicate which algorithms and operations shall be performed. This document only describes the algorithms and operations for one cipher suite, which is identified by the value 1. The mdoc indicates which cipher suite shall be used in the device engagement structure (see 9.1).

When cipher suite 1 is used, curves from Table 17 shall be used by the device retrieval security mechanisms. Support for all curves is mandatory for an mdoc reader.

The COSE_Key structures should not contain optional parameters except for the `x`, `y` and `crv` parameters. The mdoc and mdoc reader should ignore other optional parameters.

If the curve indicated in `EReaderKey` is a double-coordinate curve as specified in RFC 9053, section 7.1.1, the uncompressed form shall be used for the public key. For the public key in `EDeviceKey`, either the uncompressed or the compressed form may be used in case a double coordinate curve is used.

The brainpool curves shall have COSE Key Type EC2. COSE Key Types for other curves are specified in the IANA COSE registry.

Table 17 — Elliptic curves for cipher suite 1

Definition	Specification	Curve identifier	Purpose
Curve P-256	NIST SP 800-186	IANA COSE registry	ECDH/ECDSA
Curve P-384	NIST SP 800-186	IANA COSE registry	ECDH/ECDSA
Curve P-521	NIST SP 800-186	IANA COSE registry	ECDH/ECDSA
X25519	RFC 7748	IANA COSE registry	ECDH
X448	RFC 7748	IANA COSE registry	ECDH
Ed25519	RFC 8032	IANA COSE registry	EdDSA
Ed448	RFC 8032	IANA COSE registry	EdDSA

brainpoolP256r1	RFC 5639	IANA COSE registry	ECDH/ECDSA
brainpoolP320r1	RFC 5639	IANA COSE registry	ECDH/ECDSA
brainpoolP384r1	RFC 5639	IANA COSE registry	ECDH/ECDSA
brainpoolP512r1	RFC 5639	IANA COSE registry	ECDH/ECDSA

NOTE 1 For IANA COSE registries[2], see COSE (CBOR Object Signing and Encryption) Elliptic Curves in IANA (Internet Assigned Numbers Authority) Protocols Registries.

NOTE 2 In accordance with RFC 9052, the CDDL grammar describing a COSE_Key as used in the device retrieval security mechanisms is:

```
COSE_Key = {
   1 => int,              ; kty: key type
  -1 => int,              ; crv: EC identifier - Taken from the "COSE Elliptic Curves"
registry
  -2 => bstr,             ; x: value of x-coordinate
  ? -3 => bstr / bool     ; y: value or sign bit of y-coordinate; only applicable for EC2
key types
}
```

12.7 Validation and inspection procedures

Editor’s note: Consider adding an MSO revocation list verification procedure

12.7.1 Inspection procedure for issuer data authentication

The mdoc reader shall perform the following or functionally equivalent steps to verify that the received data is authentic and valid.

1. Validate the certificate included in the `IssuerAuth` header according to 12.7.3. See below in this clause for further requirements a VICAL as defined in Annex C is used to determine trust in the certificate in the MSO header.

2. Verify the digital signature of the IssuerAuth structure (see 12.3.4) using the working_public_key, working_public_key_parameters, and working_public_key_algorithm from the certificate validation procedure of step 1.

3. Calculate the digest value for every `IssuerSignedItem` returned in the `DeviceResponse` structure according to 12.3.5 and verify that these calculated digests equal the corresponding digest values in the MSO.

4. Verify that the `DocType` in the MSO matches the relevant `DocType` in the `Documents` structure.

5. Validate the elements in the `ValidityInfo` structure, i.e. verify that:

   — the 'signed' date is within the validity period of the certificate in the MSO header,

   — the current timestamp shall be equal or later than the ‘validFrom’ element,

   — the 'validUntil' element shall be equal or later than the current timestamp.

[2] Available at https://www.iana.org/assignments/cose/cose.xhtml.

— if the 'validUntil' element has a value later than the notAfter element in the Document Signer certificate, the mdoc reader may reject the MSO.

When a VICAL is used to determine trust in the certificate (chain) in the MSO header, the mdoc reader shall perform the following steps:

- Perform the VICAL validation procedure in C.1.8.
- Identify the `CertificateInfo` structures in the VICAL that match a certificate required to validate the certificate (chain) in the MSO header.
- For each matching `CertificateInfo` validate if the `DocType` field in the MSO matches is present in the `DocType` array in that `CertificateInfo` structure.

If there are no valid certificates according to this procedure, the VICAL used cannot be relied on to establish trust in the presented mdoc; it does not preclude relying on another VICAL, or other paths to establish trust in that mdoc.

NOTE: Trust on a VICAL means the data received is as trustworthy as per the VICAL-issuers policy. Trusting the policy is left to the mdoc verifier and is out of scope of this specification. (location of this note to be determined at the discretion of the editor)

12.7.2 Inspection procedure for mdoc authentication

The mdoc reader shall perform the following or functionally equivalent steps to verify the mdoc authentaction data. This verification procedure assumes that the issuer data authentication inspection procedure has been completed successfully.

1. If any data elements are returned as part of DeviceSigned, verify that all those data elements or their namespace are included in the keyAuthorizations map in the DeviceKeyInfo map in the MSO.
2. Verify the signature or MAC in the DeviceAuth structure (see 12.4.5 or 12.4.6) using the DeviceKey in the DeviceKeyInfo map in the MSO.

12.7.3 Inspection procedure for mdoc reader authenticationCertificate validation procedure

This subclause specifies the certification path validation procedure for certificates issued under an IACA certificate or under another CA as trust anchor. mdoc readers, mdocs and issuing authority infrastructures performing the certification path validation shall store the relevant trust anchor certificates or the information extracted from these trust anchor certificates that is required for the certification path validation in a way that preserves the availability and the authenticity of the information. Additionally, they shall have access to certificate revocation information.

This document does not mandate methods to obtain and/or to establish trust in IACA certificates and other certificates that serve as trust anchors for certification path validation. It is the responsibility of the person or organization responsible for the mdoc reader to obtain and/or to establish trust in the IACA certificates used to verify the certificates issued by the IACA, such as DS or JWS signer certificates. It is the responsibility of the issuing authority to obtain and/or to establish trust in certificates that are used as trust anchors to validate mdoc reader authentication certificates and TLS client authentication certificates.

However, an example of a method that can be used to establish such trust in IACA certificates is the VICAL as defined in Annex C, which describes a method for distribution for IACA certificates.

In any case, the issuing authority shall publicly publish its issuing authority certificate authority (IACA) certificate. This document does not prescribe methods for the generation, administration, and safekeeping of key pairs. It is the responsibility of each issuing authority to ensure that keys are generated, administered, and protected as necessary.

mdoc readers, mdocs and issuing authority infrastructures performing certification path validation shall apply the RFC 5280, section 6.1 basic path validation. Furthermore, the following steps shall be performed for certificates issued by the IACA.

— Verify that the countryName element in the subject of the IACA certificate and the countryName element in the subject of the target certificate issued under the IACA certificate are the same.

— Verify that the stateOrProvinceName element in the subject of the IACA certificate and the stateOrProvinceName element in the subject of the target certificate issued under the IACA certificate are the same if this element is present in both certificates.

If this validation succeeds it returns the final value of the working_public_key, the working_public_key_algorithm, and the working_public_key_parameters.

B.3.1 illustrates how the certification path validation can be used for an end-entity certificate issued by the IACA.

B.3.2 illustrates how the CRL validation and revocation checking can be used for an end-entity certificate issued by the IACA.

13 mDL requirements

13.1 Overview

13.2 defines the requirements for the mDL and mDL reader for the different transmission technologies for device engagement and data retrieval.

13.3 defines the requirements for the mDL, mDL reader and issuing authority for the different security mechanism.

13.4 defines the requirements for the mDL, mDL reader and issuing authority for the mDL data elements.

13.2 Transmission technologies

To ensure that device engagement is always possible, an mDL shall support at least one of the transmission technologies in Table 18. An mDL reader shall support all transmission technologies.

If the mDL supports NFC for device engagement, it shall support Static Handover, Negotiated Handover, or both, as described in 9.2. The mDL reader shall support both handover methods.

Support for session establishement during negotiated handover as defined in 9.2.3 is optional for the mdoc and mdoc reader.

Table 18 — Device engagement technologies

Transmission technology	**Further details**	**Support**		**Reference**
		mDL	mDL reader	
NFC	Negotiated handover	C[a]	M	9.2
	Static handover	C[a]	M	
QR code	-	C[a]	M	9.3

Key
C conditional
M mandatory
[a] Support for at least one of these methods is mandatory.

To ensure that data retrieval is always possible, an mDL shall support device retrieval using BLE, NFC, or both transmission technologies. An mDL reader shall support the BLE and NFC transmission technology for device retrieval.

An mDL may support Wi-Fi Aware and an mDL reader should support Wi-Fi Aware.

For device retrieval using BLE, the mDL reader shall support the mdoc central client mode and mdoc peripheral server mode, as defined in 11.1.1. The mDL and mDL reader may support the BLE L2CAP transmission modes as defined in 11.1.1.

Table gives an overview of the requirements for the mDL and mDL reader for the requirements on supported transmission technologies.

See Annex D for examples of data structures.

Table 19 — Data retrieval methods

Transmission technology	**Further details**	**Support**		**Reference**
		mDL	mDL reader	
BLE	Mdoc central client mode	C [a]	M	11.1
	Mdoc peripheral server mode	C [a]	M	
	Mdoc central l2cap mode	O	O	
	Mdoc peripheral l2cap mode	O	O	
NFC	-	C [a]	M	11.2
Wi-Fi Aware	-	O	R	11.3

Key
M mandatory
C conditional
R recommended
O optional
[a] Support for at least one of these methods is mandatory.

13.3 Security mechanisms

The certificate and CRL profile requirements in Annex B shall be applied.

All certificates issued by an IACA or another CA shall be validated according to 12.7.3.

An mDL reader needs access to the issuing authority’s certificate authority (IACA) root certificate to verify issuer data authentication. One optional method to get access to these certificates is described in Annex C.

When receiving an mdoc response, it is important that the mdoc reader verifies the business logic of the mdoc response, i.e. whether the necessary data elements are returned and whether they contain the required data. However, this document does not contain any normative requirements for such verification.

Editor's note: Consider adding an example for the previous paragraph

See Annex E for additional information on privacy and security.

Support of the ZKP mechanisms are optional for the mDL and mDL reader. When an mDL reader indicates support for one or more ZKP mechanisms in the request, it shall also accept the document if it is returned without using the ZKP mechanism.

13.4 mDL data elements

13.4.1 Document type and namespace

The `DocType` for an mDL document shall be "org.iso.18013.5.1.mDL". The number "1" in the document type might be increased in future versions of the standard.

NOTE 1 The document type field follows the following general format: [Reverse Domain].[Domain Specific Extension]. The reverse domain (org.iso) was selected to avoid collisions. This approach can be used to define other doctypes.

If the mDL reader wants to retrieve an mDL, the `DocType` field in the mdoc request shall contain the mDL document type. The use of any other value for the `DocType` field in the mdoc request is beyond the scope of this document.

The `NameSpace` for mDL data elements defined in 13.4.2 shall be "org.iso.18013.5.1". The number "1" in the namespace might be increased in future versions of the standard. Within this `NameSpace`, only data elements defined in 13.4.2 may be used. To accommodate domestic data, an issuing authority may define its own namespace, as described in 13.4.9.

NOTE 2 The namespace field follows the following general format: [Reverse Domain].[Domain Specific Extension].

NOTE 3 Since `DocType` and `NameSpace` use exact string matching, if a `DocType` or `NameSpace` uses a different version number than "1", the `DocType` or `NameSpace` is different than the ones defined in this document.

13.4.2 Data elements overview

The mDL data model is described in 13.4.1. It describes the identifier and format of the data elements specific to the mDL.

The mDL data elements shall be as defined in Table 20 and belong to namespace "org.iso.18013.5.1", see 7.1.

— The "Identifier" column is used for `DataElementIdentifier` in the device retrieval mdoc request (see 10.2).

— The "Presence" column indicates whether the presence of the element on an mDL is mandatory (M), or optional (O).

  NOTE 1 This does not mean that granting access to these elements to an mDL reader is mandatory.

— The "Encoding format" column indicates how the data elements shall be encoded. "tstr", "uint", "bstr", "bool" and "tdate" are CDDL representation types as defined in RFC 8610. This document specifies "full-date" as `full-date = #6.1004(tstr)`, where tag 1004 is specified in RFC 8943.

— In accordance with RFC 8949, section 3.4.1, a tdate data item shall contain a date-time string as specified in RFC 3339. In accordance with RFC 8943, a `full-date` data item shall contain a full-date string as specified in RFC 3339.

— The following requirements shall apply to the representation of tdate data items in mDL data elements, unless otherwise indicated:

— fraction of seconds shall not be used;

— no local offset from UTC shall be used, as indicated by setting the `time-offset` defined in RFC 3339 to “Z”.

Table 20 — Data elements

Identifier	Meaning	Definition	Presence	Encoding format
family_name	Family name	Last name, surname, or primary identifier, of the mDL holder. The value shall only use latin1[b] characters and shall have a maximum length of 150 characters.	M	tstr
given_name	Given names	First name(s), other name(s), or secondary identifier, of the mDL holder. The value shall only use latin1[b] characters and shall have a maximum length of 150 characters.	M	tstr
birth_date	Date of birth	Day, month and year on which the mDL holder was born. If unknown, approximate date of birth	M	full-date
issue_date	Date of issue	Date when mDL was issued	M	tdate or full-date
expiry_date	Date of expiry	Date when mDL expires	M	tdate or full-date
issuing_country	Issuing country	Alpha-2 country code, as defined in ISO 3166-1, of the issuing authority’s country or territory	M	tstr
issuing_authority	Issuing authority	Issuing authority name. The value shall only use latin1[b] characters and shall have a maximum length of 150 characters.	M	tstr
document_number	Licence number	A number assigned or calculated by the issuing authority. What number is used is out of scope of this specification. The value shall only use latin1[b] characters and shall have a maximum length of 150 characters.	M	tstr
portrait	Portrait of mDL holder	A reproduction of the mDL holder’s portrait. See 13.4.3	M	bstr

Identifier	Meaning	Definition	Presence	Encoding format
driving_privileges	Categories of vehicles/ restrictions/ conditions	Driving privileges of the mDL holder. See 13.4.5	M	See 7.2.4
un_distinguishing_sign	UN distinguishing sign	Distinguishing sign of the issuing country according to ISO/IEC 18013-1:2018, Annex F. If no applicable distinguishing sign is available in ISO/IEC 18013-1, an IA may use an empty identifier or another identifier by which it is internationally recognized. In this case the IA should ensure there is no collision with other IA's.	M	tstr
administrative_number	Administrative number	An audit control number assigned by the issuing authority. The value shall only use latin1[b] characters and shall have a maximum length of 150 characters.	O	tstr
sex	Sex	mDL holder's sex using values as defined in ISO/IEC 5218 with the following change: the meaning of the value 9 is defined as "not specified"	O	uint
height	Height (cm) [a]	mDL holder's height in centimetres	O	uint
weight	Weight (kg) [a]	mDL holder's weight in kilograms	O	uint
eye_colour	Eye colour	mDL holder's eye colour. The value shall be one of the following: "black", "blue", "brown", "dichromatic", "grey", "green", "hazel", "maroon", "pink", "unknown".	O	tstr
hair_colour	Hair colour	mDL holder's hair colour. The value shall be one of the following: "bald", "black", "blond", "brown", "grey", "red", "auburn", "sandy", "white", "unknown".	O	tstr
birth_place	Place of birth	Country and municipality or state/province where the mDL holder was born. The value shall only use latin1[b] characters and shall have a maximum length of 150 characters.	O	tstr
resident_address	Permanent place of residence	The place where the mDL holder resides and/or may be contacted (street/house number, municipality etc.). The value shall only use latin1[b] characters and shall have a maximum length of 150 characters.	O	tstr

Identifier	Meaning	Definition	Presence	Encoding format
portrait_capture_date	Portrait image timestamp	Date when portrait was taken	O	tdate
age_in_years	Age attestation: How old are you (in years)?	The age of the mDL holder	O	uint
age_birth_year	Age attestation: In what year were you born?	The year when the mDL holder was born	O	uint
age_over_NN	Age attestation: Nearest “true” attestation above request or “false” attestation below request	See 13.4.6	O	bool
issuing_jurisdiction	Issuing jurisdiction	Country subdivision code of the jurisdiction that issued the mDL as defined in ISO 3166-2:2020, Clause 8. The first part of the code shall be the same as the value for issuing_country.	O	tstr
nationality	Nationality	Nationality of the mDL holder as a two letter country code (alpha-2 code) defined in ISO 3166-1	O	tstr
resident_city	Resident city	The city where the mDL holder lives. The value shall only use latin1[b] characters and shall have a maximum length of 150 characters.	O	tstr
resident_state	Resident state/province/district	The state/province/district where the mDL holder lives. The value shall only use latin1[b] characters and shall have a maximum length of 150 characters.	O	tstr
resident_postal_code	Resident postal code	The postal code of the mDL holder. The value shall only use latin1[b] characters and shall have a maximum length of 150 characters.	O	tstr
resident_country	Resident country	The country where the mDL holder lives as a two letter country code (alpha-2 code) defined in ISO 3166-1.	O	tstr
biometric_template_xx	Biometric template XX	See 13.4.7	O	bstr
family_name_national_character	Family name in national characters	The family name of the mDL holder using full UTF-8 character set.	O	tstr
given_name_national_character	Given name in national characters	The given name of the mDL holder using full UTF-8 character set.	O	tstr
signature_usual_mark	Signature / usual mark	Image of the signature or usual mark of the mDL holder, see 713.4.8	O	bstr

Identifier	Meaning	Definition	Presence	Encoding format
Key Presence: M mandatory O optional				
NOTE 1 The ‘UN Distinguishing sign’ element is added for purposes of the UN conventions[26][27] on driving licences. NOTE 2 The ‘sex’ element is equal to the ‘gender’ fields in ISO/IEC 18013-1 and ISO/IEC 18013-2. NOTE 3 The ‘issuing jurisdiction’ element can be used in cases where the issuing jurisdiction is different from the issuing authority or the issuing country. [a] The mDL reader can convert to the local unit of measurement. [b] Latin1 shall be as defined in ISO/IEC 8859-1 as Latin alphabet No. 1.				

If any data element is returned to the mDL reader, then the information necessary to verify that the person presenting the mDL is the mDL holder shall be returned to the mDL reader if that information is requested.

NOTE 2 From the set of data elements with mandatory presence in column "Presence" from Table 5, the portrait of the mDL holder is the only data element for verifying that the person presenting the mDL is the mDL holder.

An mDL may require mdoc reader authentication (see 12.5) before releasing data elements not marked as mandatory in Table 20. An mDL shall not require mdoc reader authentication as a precondition for returning any of the mandatory data elements. An mDL may offer functionality to the mDL holder to pre-authorize the return of mandatory data elements selected by the mDL holder to mDL readers using mdoc reader authentication.

Editor’s note: Clarify how the requirement for not requiring mdoc reader authentication applies in the case of an incorrect signature

NOTE 3 The intention of these requirements is that the mDL holder is always able to use the mDL as a driving licence if the mDL holder chooses to do that, including if an mDL reader does not use mdoc reader authentication.

If the mDL reader retrieved the “issuing_country” data element, it shall verify that the value of that element matches the `countryName` element in the subject field within the DS certificate (see Annex B).

If the mDL reader retrieved the “issuing_jurisdiction” data element, it shall verify that the value of that element matches the `stateOrProvinceName` element in the subject field within the DS certificate (see Annex B). This is only required if the `stateOrProvinceName` element is present in the DS certificate.

NOTE 5 The mandatory presence of the `countryName` element and the optional presence of the `stateOrProvinceName` element in the DS or JWS certificate can provide the contents of those fields to the mDL reader, regardless of whether their respective data elements, “issing_country” and “issuing_jurisdiction” were returned to the mDL reader.

Any element in Table 20 and any domestic data element (see 13.4.9) that is returned by the mDL shall be returned in either `IssuerSignedItems` or `DeviceSignedItems`. As specified in 12.3.4, to be able to return some or all of its data elements as `DeviceSignedItems`, an mDL has to be authorized by the issuing authority. An issuing authority shall not authorize the mDL to use `DeviceSignedItems` for any data element in Table 20 unless the security level of the mdoc's secure area is equivalent to or higher than the security level of the issuing authority’s infrastructure used to issue MSOs. The intent of this requirement is that this mechanism does not decrease the security of the solution. In particular, a security certification of the mdoc's secure area should be performed, followed by security assessments at regular intervals. For further recommendations on the support of `IssuerSignedItems` and `DeviceSignedItems` see E.6.

NOTE 6 The support of DeviceSignedItems for data elements in Table 20 is a change compared to the provisions in ISO/IEC 18013-5:2021. In ISO/IEC 18013-5:2021 it is not allowed to use DeviceSignedItems for any data element from table 20 (table 5 in ISO/IEC 18013-5:2021). Issuing authorities considering issuing mDLs which support DeviceSignedItems for these data elements should take into consideration that already deployed ISO/IEC 18013-5 conformant mDL readers potentially do not support this change, which could lead to mDL readers rejecting these mDLs. Further noting that updating the mDL readers will take time.

NOTE 7 Issuing authorities have the responsibility to ensure that controls to prevent the unintended return of mDL data are implemented, including providing mDL holders with the means to control what data is returned.

13.4.3 Portrait of mDL holder

The portrait of mDL holder consists of one portrait image and shall follow the requirements on the face image as specified in ISO/IEC 18013-2:2020, Annex D. One of the following image formats shall be used: JPEG or JPEG2000. The image data shall be encoded as binary data.

13.4.4 Issuing authority

The issuing authority element identifies the administrative authority entitled to issue the driving licence, or the issuing country if separate licensing authorities have not been authorized. The issuing authority element is represented by a string.

NOTE The contents of this field correspond to the contents of the issuing authority element on the IDL, and can indicate a local, regional, or national organisation. Note that, like in ISO/IEC 18013-1, the term issuing authority can also refer to a central government agency, acting on behalf of multiple local or regional issuing authorities.

13.4.5 Categories of vehicles/restrictions/conditions

The categories of vehicles/restrictions/conditions contain information describing the driving privileges of the mDL holder. The definition of the elements in the DrivingPrivilege structure can be found in ISO/IEC 18013-1:2018, Clause 5. The possible values for the elements are defined in ISO/IEC 18013-1:2018, Annex B and ISO/IEC 18013-2:2020, Annex A.

NOTE 1 Regardless of driving privileges' status, other mDL data elements can be used for identification purposes as described by this document.

The driving privileges structure shall be encoded as CBOR and shall be formatted as follows:

```
DrivingPrivileges = [
    * DrivingPrivilege
]

DrivingPrivilege = {
    "vehicle_category_code" : tstr      ; Vehicle category code as per ISO/IEC
18013-1 Annex B
    ? "issue_date" : full-date          ; Date of issue encoded as full-date
    ? "expiry_date" : full-date         ; Date of expiry encoded as full-date
    ? "codes" : [+Code]                 ; Array of code info
    * tstr => RFU
}

Code = {
    "code": tstr                        ; Code as per ISO/IEC 18013-2 Annex A
    ? "sign": tstr                      ; Sign as per ISO/IEC 18013-2 Annex A
    ? "value": tstr                     ; Value as per ISO/IEC 18013-2 Annex A
    * tstr => RFU
}
```

NOTE 2 The `DrivingPrivileges` structure can be an empty array.

NOTE 3 mDL readers displaying vehicle category code information using pictographs (as per the 1968 UN Vienna Convention on Road Traffic, as amended) can consult ISO/IEC 18013-1 for the pictographs to use.

`full-date` is defined in 13.4.2.

An example can be found in D.2.1.

13.4.6 Age attestation: nearest "true" attestation above request

Age_over_NN is a set of data elements used to communicate to an mDL verifier, in a data-minimized manner, whether the mDL holder is as old or older than a specified age, or if the mDL holder is younger than a specified age. An age attestation identifier has the format age_over_NN, where NN is used to encode the age value, with N being a digit from 0 to 9. Consequently, the age attestation identifier can encode any value from "age_over_00" to "age_over_99". The value of an age attestation identifier can be TRUE or FALSE.

When an age attestation data element is returned by the mDL with a value of TRUE, it means the age of the mDL holder is equal or higher than the age in the data element identifier. When an age attestation data element is returned by the mDL holder with a value of FALSE, it means the age of the mDL holder is lower than the age in the data element identifier.

If an mDL verifier includes age_over_NN in a request, it has the meaning of "provide the nearest age attestation equal to or larger than NN with value TRUE, or smaller than NN with value FALSE". More specifically, after receiving an age_over_NN request, the logic to determine the appropriate response shall be equivalent to the following.

1. For all age attestations data elements of the form age_over_nn (lowercase nn is used here to describe the age of the data element identifiers stored on the mDL) stored on the mDL, consider all the attestations with value TRUE. From among these attestations, check if an attestation exists where nn is equal to or larger than NN. If one and only one such attestation exists, this is the response. If more than one such attestation exists, the response shall be the attestation with the smallest difference between nn and NN.

2. If step 1 does not produce a response, for all age attestations of the form age_over_nn stored on the mDL, consider all the attestations with value FALSE. From among these attestations, check if an attestation exists where nn is equal to or smaller than NN. If one and only one such attestation exists, this is the response. If more than one such attestation exists, the response shall be the attestation with the smallest difference between NN and nn.

3. If step 2 does not produce a response, no age_over_NN data element shall be returned.

For the purposes of the additional request info in 10.2.4, a request for a data element is considered satisfied if any age_over_NN data element is returned according to the logic described above.

In case of device retrieval, the value of an age_over_NN data element shall be calculated by the issuing authority infrastructure to be valid at the value of the timestamp in the `validFrom` element in the MSO from 12.3.4.

During a single data retrieval phase (see Figure 3), an mDL reader shall not request more than two age_over_NN data elements, and an mDL should not return more than two age_over_NN data elements.

Examples of requests and responses and their meaning can be found in D.2.2.

NOTE 1 Including more rather than less age_over_NN statements in an mDL has the following consequences, both of which are beneficial from a privacy perspective.

1. If an mDL does not respond to an age_over_NN request, the most likely next step of a verifier will be to ask for the mDL holder's date of birth. This would be the only remaining way in which the original business question could be answered. Including additional age_over_NN statements decreases the probability of a no response.

2. Given the question "are you older than 18?", the response "I am older than 21" and the response "I am older than 60" would both answer the question. However, the response "I am older than 21" is considered better from a privacy point of view, since the statement is closer to the request. Including additional age_over_NN statements will on average decrease the difference between the requested age and the response age.

Including more rather than less age_over_NN statements in a mDL on the other hand could allow a verifier to gain a more accurate estimate of a person's true age, if for whatever reason such a verifier chooses not to ask directly for the mDL holder's date of birth. This would require repeated requests.

NOTE 2 A request can include two age_over_NN statements to support effectively asking whether the age of the mDL holder falls within a certain range.

NOTE 3 It is possible that an mDL does not have an age_over_NN element available to respond to the request from the mDL reader. Using AlternativeDataElements (see 10.2.4) the mDL reader has the option of indicating that there are different data elements that fulfil the business requirements that it has. Examples of these are 'age_in_years' and 'birth_date'.

NOTE 4 Because the `validFrom` element in the MSO is used for the validity of the age_over_NN statements, it is possible to issue mDLs with age_over_NN statetements that are only valid in the future.

13.4.7 Biometric template

This element contains optional facial, fingerprint, iris, or other biometric information of the mDL holder. Biometric information is encoded according to the biometric template defined in ISO/IEC 18013-2:2020, Table 7 and C.4.7. However, the first tag in ISO/IEC 18013-2:2020, Table C.11 ('75' for Facial, '63' for Finger, '76' for Iris, etc) shall be omitted. The value of the biometric_template_xx data element shall therefore start with tag '7F 61'. A biometric template identifier has the format biometric_template_xx where xx shall be replaced with the corresponding "Abstract value name" found in ISO/IEC 19785-3:2020, Table 7, according to the following convention: capitalized characters are replaced with their lowercase equivalent and spaces or non-alphanumeric characters are replaced by underscores (_).

EXAMPLE The "FACE" template corresponds to "biometric_template_face" and the "SIGNATURE/SIGN" template corresponds to "biometric_template_signature_sign".

If the biometric_template_face is used, the biometric data block (tag '5F 2E') shall contain a JPEG or JPEG2000 format, as specified in 13.4.3.

13.4.8 Signature or usual mark

The signature or usual mark of the mDL holder consists of one image. One of the following image formats shall be used: JPEG or JPEG2000. The image data shall be encoded as binary data.

13.4.9 Domestic data elements

Domestic data are data elements which are not specified in this document. An issuing authority may specify its own data elements within its namespace. Since the namespace for mDL data in this document is "org.iso.18013.5.1", the issuing authority infrastructure should use country-specific or issuer-specific namespaces by appending the ISO 3166-1 alpha-2 country code or the ISO 3166-2 region code after a period.

EXAMPLE The United States namespace is "org.iso.18013.5.1.US" and the Iowa namespace is "org.iso.18013.5.1.US-IA".

13.4.10 Country codes

ISO 3166-1 shall be used as a source for country code identifiers. If no applicable country code is available in ISO 3166-1, an IA may use one of the user-assigned country code elements, as indicated in ISO 3166-1 or another identifier by which it is internationally recognized. In these cases, the IA should ensure there is no collision with other IAs. These provisions apply to all occurrences of country code identifiers in this document unless stated otherwise.

13.5 Commen recognition

An mDL verifier can use the logo in Figure 5, including in association with local signage and branding, to indicate that it accepts mDLs. It can also be used for other mdoc based documents.

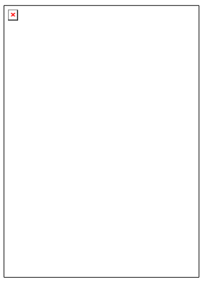

Figure 5 — Logo

When scaled, the x to y dimension ratio shall be retained. Rendering the logo in different colours and with different backgrounds is allowed. However, such changes should be applied with care to maintain common recognition, especially across jurisdictional boundaries.

This logo is intended for readers that are implementing the requirements from ISO/IEC 18013-5; support for clause 13.4 and Annex B do not apply when reading other mdoc based documents. A way to verify if the requirements from ISO/IEC 18013-5 have been met is to undergo testing/certification under ISO/IEC 18013-6.

Example 1

Figure 6 and Figure 7 provide examples of how relying parties could use the logo to convey their acceptance of mDLs.

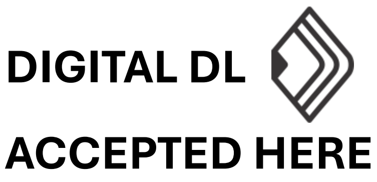

Figure 6

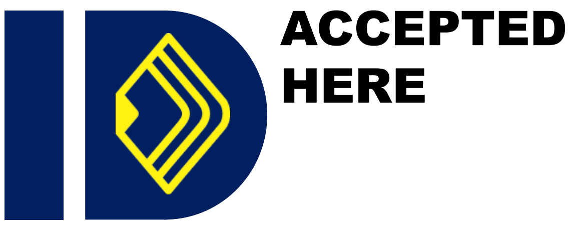

Figure 7

To help mDL holders associate the logo with acceptance of their mDL, issuing authorities and wallets can include the logo in their mDL branding.

Example 2

Figure 8 provides an example of how an mDL issuer could incorporate the logo into their branding.

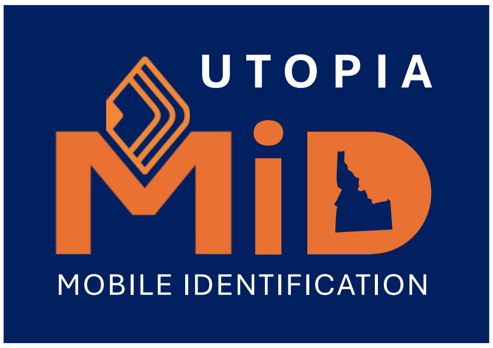

Figure 8

Annex A
(informative)

BLE L2CAP transmission profile

Editor’s note: The contents of this section has been moved to the main body

Annex B
(normative)

Certificate and CRL profiles

B.1 Certificate profiles

B.1.1 Overview

The certificate profiles defined in this annex are mandatory for mDLs, mDL readers and mDL issuing authority infrastructures, except for the mdoc reader authentication and TLS client certificate profiles in B.1.7 and B.1.8, which are recommended.

The IACA root certificate is the root certificate used to issue all end-entity certificates, except possibly mdoc reader authentication and TLS client authentication certificates. The allowed usage of the end-entity certificates is defined using the extended key usage extension. All end-entity certificates signed by an IACA root certificate shall contain the same country code as the IACA certificate. An IACA certificate may contain the stateOrProvinceName element to indicate that this IACA certificate only issues mDLs within a particular state or province. If the element is present in an IACA certificate, it shall also be present and have the same value in the end-entity certificates signed by that IACA certificate.

The IACA root certificate shall use the IACA root certificate profile as defined in B.1.2.

If an IACA link certificate is created, it shall use the IACA link certificate profile as defined in B.1.3.

For issuer data authentication (see 9.1.2), the issuing authority shall use the mDL document signer certificate profile as defined in B.1.4. This certificate is included in the `x5chain` element of `IssuerAuth` (see 9.1.2.4). The IACA root certificate shall not be included in the `x5chain` element.

NOTE 1 If the `x5chain` contains multiple certificates, the first element in the `x5chain` array is the mDL DS certificate. mdoc readers using only IACA root certificates for validation of DS certificates do not need to parse any element beyond the first in the `x5chain` array.

For JWS, the issuing authority shall use the JWS signer certificate profile as defined in B.1.5. This certificate is included in the `x5c` parameter of the JWS (see 9.2.2). The IACA root certificate shall not be included in the `x5c` parameter.

NOTE 2 If the `x5c` parameter contains multiple certificates, the first element in the `x5c` array is the JWS signer certificate. mdoc readers using only IACA root certificates for validation of JWS signer certificates do not need to parse any element beyond the first in the `x5c` array.

For mdoc reader authentication (see 9.1.4), mDLs and mDL readers should use the mdoc reader authentication certificate profile as defined in B.1.7.

If a certificate that is issued by an IACA root certificate indicates support for OCSP, the OCSP signer certificate shall comply with the OCSP signer certificate profile as defined in B.1.9.

For MSO revocation (see 12.3.6) the certificate used to sign the identifier list or status list should comply with the certificate profile as defined in B.1.10.

The CRL indicated in an IACA root certificate, an IACA link certificate and any certificate signed by an IACA root certificate shall comply with the requirements in B.2.

All certificates shall be DER encoded.

For each certificate profile, the presence column indicates whether an element is mandatory (M), optional (O) or conditional (C). The criticality column indicates whether an element is critical (C) or non-critical (NC).

The following extensions shall not be used:

— PolicyMappings,

— NameConstraints,

— PolicyConstraints,

— InhibitAnyPolicy,

— FreshestCRL.

Certificates in this annex use OIDs for extended key usage extension. The OIDs have the following definition:

— id-mdl OBJECT IDENTIFIER ::= { iso(1) standard(0) 18013 5 },

— id-mdl-kp OBJECT IDENTIFIER ::= { id-mdl 1 } - - arc for extended key purposes,

— id-mdl-kp-mdlDS OBJECT IDENTIFIER ::= { id-mdl-kp 2 } - - arc for mDL DS,

— id-mdl-kp-mdlReaderAuth OBJECT IDENTIFIER ::= { id-mdl-kp 6 } - - arc for mdoc reader authentication used by mDL readers,

— id-mdl-kp-mdlIACALink OBJECT IDENTIFIER ::= { id-mdl-kp 4 } - - arc for mDL IACA Link,

— id-mdl-kp-mdlIACA OBJECT IDENTIFIER ::= { id-mdl-kp 7} – arc for mDL IACA.

ISO/IEC 23220-4 specifies OIDs that are not mDL specific. These are the following OIDs:

— id-mdoc OBJECT IDENTIFIER ::= { iso(1) standard(0) 23220 4 },

— id-mdoc-kp OBJECT IDENTIFIER ::= { id-mdoc 1 } -- arc for key purposes

— id-mdoc-kp-mdocDS OBJECT IDENTIFIER ::= { id-mdoc-kp 2 } – arc for document Signer

— id-mdoc-kp-mdocIACALink OBJECT IDENTIFIER ::= { id-mdoc-kp 4 } – arc for IACA Link

— id-mdoc-kp-mdocReaderAuth OBJECT IDENTIFIER ::= { id-mdoc-kp 6 } – arc for Reader Authentication certificate

— id-mdoc-kp-mdocIACA OBJECT IDENTIFIER ::= { id-mdoc-kp 7 } – arc for IACA

— id-mdoc-kp-mdocVICAL OBJECT IDENTIFIER ::= { id-mdoc-kp 8 } – arc for VICAL Signer

These OIDs can be used in conjunction with the mDL specific OIDs. It is recommended for mdocs, mdoc readers and issuing authority infrastructure to use these OIDs and to be able to interpret them. Note that when using these generic OIDs, it is still mandatory for the mDL specific OIDs to also be present. This is done to ensure backwards compatibility with ISO/IEC 18013-5:2021.

B.1.2 IACA root certificate

This certificate profile defines the IACA root certificate, establishing the allowed security parameters for interoperability. The IACA root certificate is used as the root for all certificates defined in this annex. One IACA root certificate may be used by multiple issuing authorities within one country. See Table B.1 for details.

Issuing authorities should define the validity period of the IACA root certificate as the sum of:

— the longest validity length of the end entity certificates to be issued (e.g. document signer certificates or OCSP signer certificates);

— usage period: time during which end entity certificates will be issued;

— lead times: time required to create and disseminate the IACA root certificate before its usage period.

NOTE The longest validity length of the end entity certificates includes any document signer certificates for IDLs with a secure integrated circuit. IDLs with a secure integrated circuit and the corresponding document signer certificates typically have a longer maximum validity than document signer certificates for mDLs.

The private key usage period shall be carefully set, balancing the risk of having too many documents issued under the same IACA root certificate against the efforts and lead time required to create new IACA root certificates. The recommended period is between 3 to 5 years.

This document defines the IACA certificate as a self-signed root certificate for optimized performance and maximum cross-border interoperability.

The IACA public key can additionally be used in other PKI schemes, e.g. signed by a top root national CA or cross-signed by another authority. Those additional uses and the distribution mechanisms are outside the scope of this document.

Editor’s note: There is an open item to add an optional field for the MSO revocation list aggregation URI. Proposals are requested for addition of the field

Table B.1 — IACA root certificate

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Version	4.1.2.1	M		Shall be `v3`.
Serial number	4.1.2.2	M		Non-sequential positive, non-zero integer, shall contain at least 63 bits of output from a CSPRNG, should contain at least 71 bits of output from a CSPRNG, maximum 20 octets.
Signature	4.1.2.3	M		Value shall match the OID in the signature algorithm (below).

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Issuer	4.1.2.4	M		`countryName` is mandatory. The value shall be in upper case and contain the ISO 3166-1 alpha-2 code of the issuing country, exactly the same value as in the issuing country data element. The `countryName` shall be `PrintableString`. `stateOrProvinceName` is optional. If this element is present, the element shall also be present in the end-entity certificates and hold the same value. The value shall exactly match the value of the data element "issuing_jurisdiction", if that element is present on the mDL. `organizationName` is optional. Its value is at the discretion of the IACA. `commonName` shall be present. Its value is at the discretion of the IACA. `serialNumber` is optional. If present, it shall be a `PrintableString`. Attributes that have a `DirectoryString` and for which the encoding is not listed above syntax shall be either `PrintableString` or `UTF8String`.
Validity	4.1.2.5	M		
notBefore		M		Date on which the certificate validity period begins.
notAfter		M		Maximum of 20 years after "notBefore" date. NOTE The 20-year validity period results from the possibility of using the IACA root certificate for issuing an IDL according to ISO/IEC 18013-3, which allows the use of DS certificates with validity periods up to 15 years. If the IACA root certificate is only used to issue mDLs, a maximum validity period of 9 years is sufficient.
Subject	4.1.2.6	M		Same exact binary value as Issuer.
Subject public key info	4.1.2.7	M		
algorithm		M		`1.2.840.10045.2.1` (Elliptic curve)
parameters		M		Implicitly specify curve parameters through an OID associated with one of the following curves specified in NIST SP 800-186: `1.2.840.10045.3.1.7` (Curve P-256) `1.3.132.0.34` (Curve P-384) `1.3.132.0.35` (Curve P-521) Or one of the following curves specified in RFC 5639: `1.3.36.3.3.2.8.1.1.7` (brainpoolP256r1) `1.3.36.3.3.2.8.1.1.9` (brainpoolP320r1) `1.3.36.3.3.2.8.1.1.11` (brainpoolP384r1) `1.3.36.3.3.2.8.1.1.13` (brainpoolP512r1)
subjectPublicKey		M		Public key shall be encoded in uncompressed form.
X.509v3 extensions	4.2	M		Further extensions may be present if they are marked non-critical.
Subject key identifier	4.2.1.2	M	NC	SHA-1 hash of the subject public key `BIT STRING` value (excluding tag, length, and number of unused bits).
Key usage	4.2.1.3	M	C	
Digital signature				`0`
Non-repudiation				`0`
Key encipherment				`0`

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Data encipherment				0
Key agreement				0
Key certificate signature				1
CRL signature				1
Encipher only				0
Decipher only				0
Issuer alternative name	4.2.1.7	M	NC	The issuer alternative name extension shall provide contact information for the issuer of the certificate. For that purpose, the issuer alternative name shall include at least one of — `rfc822Name`, or — `uniformResourceIdentifier`. NOTE This contact information is intended to help establish trust in the certificate and the certified key by appropriate out of band mechanisms. Note that this information is only meant for contact information and does not in itself imply any level of trust in the certificate.
Basic constraints	4.2.1.9	M	C	
CA		M		TRUE
pathLenConstraint		M		0
CRLDistributionPoints	4.2.1.13	O	NC	The ‘reasons’ and ‘cRL Issuer’ fields shall not be used. Presence of this field is not recommended; its presence is allowed for backwards compatibility reasons.
distributionPoint		C		URI for CRL distribution point
Signature algorithm	4.1.1.2	M		Options: `1.2.840.10045.4.3.2` (ECDSA-with SHA256) `1.2.840.10045.4.3.3` (ECDSA-with SHA384) `1.2.840.10045.4.3.4` (ECDSA with SHA512)
Signature value	4.1.1.3	M		Value according to the signature algorithm. By creating this signature, the CA certifies the binding between the public key material and the subject of the certificate, i.e. the IACA.
Key Presence: M mandatory O optional Criticality: C critical NC not critical				

B.1.3 IACA link certificate

This certificate profile defines the IACA link certificate, establishing the allowed security parameters for interoperability. The IA should generate and distribute an IACA link certificate when doing an IACA re-key. The link certificate establishes a trust path from the old IACA root certificate to the new one. See Table B.2 for details.

Table B.2 — IACA link certificate

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Version	4.1.2.1	M		Shall be `v3`.
Serial number	4.1.2.2	M		Non-sequential positive, non-zero integer, shall contain at least 63 bits of output from a CSPRNG, should contain at least 71 bits of output from a CSPRNG, maximum 20 octets.
Signature	4.1.2.3	M		Value shall match the OID in the signature algorithm (below).
Issuer	4.1.2.4	M		Same exact binary value as the subject in the old IACA root certificate, for which the respective private key is signing this link certificate.
Validity	4.1.2.5	M		
notBefore		M		Date on which the link certificate validity period begins.
notAfter		M		Date shall not be after the “notAfter” date of the old IACA root certificate.
Subject	4.1.2.6	M		Same exact binary value as Issuer in the new IACA root certificate.
Subject public key info	4.1.2.7	M		Same as the subject public key info in the new IACA root certificate.
X.509v3 extensions	4.2	M		Further extensions may be present if they are marked non-critical.
Authority key identifier	4.2.1.1	M	NC	
keyIdentifier		M		Same value as the subject key identifier of the old IACA root certificate
Subject key identifier	4.2.1.2	M	NC	SHA-1 hash of the subject public key `BIT STRING` value (excluding tag, length, and number of unused bits).
Key usage	4.2.1.3	M	C	
Digital signature				`0`
Non-repudiation				`0`
Key encipherment				`0`
Data encipherment				`0`
Key agreement				`0`
Key certificate signature				`1`
CRL signature				`1`
Encipher only				`0`
Decipher only				`0`
Subject alternative name	4.2.1.6	C	NC	The presence is conditional. If the IACA wants to change the DN when doing a CA rollover, the subject alternative name extension shall include a `directoryName` entry with the new Issuer DN.
Issuer alternative name	4.2.1.7	M	NC	The issuer alternative name extension shall provide contact information for the issuer of the certificate. For that purpose, the issuer alternative name shall include at least one of — `rfc822Name`, or — `uniformResourceIdentifier`. NOTE This contact information is intended to help establish trust in the certificate and the certified key by appropriate out of band mechanisms. Note that this information is only meant for contact information and does not in itself imply any level of trust in the certificate.

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Basic constraints	4.2.1.9	M	C	
CA		M		`TRUE`
pathLenConstraint		M		`0`
CRLDistributionPoints	4.2.1.13	M	NC	The ‘reasons’ and ‘cRL Issuer’ fields shall not be used.
distributionPoint		M		URI for CRL distribution point
Signature algorithm	4.1.1.2	M		Options: `1.2.840.10045.4.3.2` (ECDSA-with SHA256) `1.2.840.10045.4.3.3` (ECDSA-with SHA384) `1.2.840.10045.4.3.4` (ECDSA with SHA512)
Signature value	4.1.1.3	M		Value according to the signature algorithm. By creating this signature, the CA certifies the binding between the public key material and the subject of the certificate, i.e. the Public Key and DN of new IACA root certificate.
Key Presence: M mandatory O optional C conditional Criticality: C critical NC not critical				

B.1.4 Document signer certificate

This certificate is used to sign the mobile security object in the device retrieval mdoc response.

Additional information on privacy and security can be found in E.7.

NOTE A method to enhance its security is for the issuing authority to use certificates that have a short lifespan (for example, a few weeks), and if a compromise is detected, to revoke the document signer certificate and reissue and disseminate to all concerned mDL updated data signed with a new document signer certificate. See Table B.3 for details.

Table B.3 — Document signer certificate

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Version	4.1.2.1	M		Shall be `v3`.
Serial number	4.1.2.2	M		Non-sequential positive, non-zero integer, shall contain at least 63 bits of output from a CSPRNG, should contain at least 71 bits of output from a CSPRNG, maximum 20 octets.
Signature	4.1.2.3	M		Value shall match the OID in the signature algorithm (below).
Issuer	4.1.2.4	M		Same exact binary value as the subject of IACA certificate.
Validity	4.1.2.5	M		
notBefore		M		Date on which the certificate validity period begins.
notAfter		M		Maximum of 457 days after “notBefore” date

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Subject	4.1.2.6	M		`countryName` is mandatory. The value shall be in upper case and contain the ISO 3166-1 alpha-2 code of the issuing country, exactly the same value as in the issuing country data element. The `countryName` shall be `PrintableString`. `stateOrProvinceName` is optional. If this element is present in the IACA root certificate, this element shall be present and hold the same value. The value shall exactly match the value of the data element "issuing_jurisdiction", if that element is present on the mDL. `organizationName` is optional. Its value is at the discretion of the IACA. `commonName` shall be present. Its value is at the discretion of the IACA. `localityName` is optional. Its value is at the discretion of the IACA. `serialNumber` is optional. If present, it shall be a `PrintableString`. Attributes that have a `DirectoryString` and for which the encoding is not listed above syntax shall be either `PrintableString` or `UTF8String`.
Subject public key info	4.1.2.7	M		

algorithm		M		If any of the curves specified below for the `parameters` field is used, the following OID must be used, as specified in RFC 5480 and RFC 5639:
				`1.2.840.10045.2.1` (id-ecPublicKey)
				For curves Ed25519 or Ed448, one of the following OIDs must be used, as specified in RFC 8410:
				`1.3.101.112(Curve Ed25519)`
				`1.3.101.113(Curve Ed448)`

parameters		C		This field must only be present when the `algorithm` field contains the OID `1.2.840.10045.2.1`. Implicitly specify curve parameters through an OID associated with one of the following curves specified in NIST SP 800-186: `1.2.840.10045.3.1.7` (Curve P-256) `1.3.132.0.34` (Curve P-384) `1.3.132.0.35` (Curve P-521) Or one of the following curves specified in RFC 5639: `1.3.36.3.3.2.8.1.1.7` (brainpoolP256r1) `1.3.36.3.3.2.8.1.1.9` (brainpoolP320r1) `1.3.36.3.3.2.8.1.1.11` (brainpoolP384r1) `1.3.36.3.3.2.8.1.1.13` (brainpoolP512r1)
subjectPublicKey		M		For all curves except Ed25519 or Ed448, the public key shall be encoded in uncompressed form.
X.509v3 extensions	4.2	M		Further extensions may be present if they are marked non-critical.
Authority key identifier	4.2.1.1	M	NC	
keyIdentifier		M		Same value as the subject key identifier of the IACA root certificate
Subject key identifier	4.2.1.2	M	NC	SHA-1 hash of the subject public key `BIT STRING` value (excluding tag, length, and number of unused bits).

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Key usage	4.2.1.3	M	C	
Digital signature				`1`
Non-repudiation				`0`
Key encipherment				`0`
Data encipherment				`0`
Key agreement				`0`
Key certificate signature				`0`
CRL signature				`0`
Encipher only				`0`
Decipher only				`0`
Subject alternative name	4.2.1.6	O	NC	
Issuer alternative name	4.2.1.7	M	NC	The issuer alternative name extension shall provide contact information for the issuer of the certificate. For that purpose, the issuer alternative name shall include at least one of — `rfc822Name`, or — `uniformResourceIdentifier`. NOTE This contact information is intended to help establish trust in the certificate and the certified key by appropriate out of band mechanisms. Note that this information is only meant for contact information and does not in itself imply any level of trust in the certificate.
Extended key usage	4.2.1.12	M	C	
Key usage		M		Mandatory: `1.0.18013.5.1.2` (mdlDS) Recommended: `1.0.23220.4.1.2` (mdocDS)
CRLDistributionPoints	4.2.1.13	M	NC	The ‘reasons’ and ‘cRL Issuer’ fields shall not be used.
distributionPoint		M		URI for CRL distribution point
Signature algorithm	4.1.1.2	M		Options: `1.2.840.10045.4.3.2` (ECDSA-with SHA256) `1.2.840.10045.4.3.3` (ECDSA-with SHA384) `1.2.840.10045.4.3.4` (ECDSA with SHA512)
Signature value	4.1.1.3	M		Value according to the signature algorithm. By creating this signature, the CA certifies the binding between the public key material and the subject of the certificate, i.e. the document signer.

Key

Presence:

M mandatory

O optional

Criticality:

C critical

NC not critical

B.1.5 mdoc reader authentication

The mDL reader should use the certificate profile according to Table B.6 for mdoc reader authentication (see 9.1.4).

Table B.6 — mdoc reader authentication

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Version	4.1.2.1	M		Shall be `v3`.
Serial number	4.1.2.2	M		Non-sequential positive, non-zero integer, shall contain at least 63 bits of output from a CSPRNG, should contain at least 71 bits of output from a CSPRNG, maximum 20 octets.
Signature	4.1.2.3	M		Value shall match the OID in the signature algorithm (below).
Issuer	4.1.2.4	M		The same binary value as the Subject of a CA certificate used for mdoc reader authentication. NOTE 1 This CA certificate, and the manner in which it is trusted by an mDL, is outside the scope of this document.
Validity	4.1.2.5	M		
notBefore		M		Date on which the certificate validity period begins.
notAfter		M		Maximum of 1 187 days after “notBefore” date
Subject	4.1.2.6	M		`commonName` shall be present. Other elements may be present in the Subject field.
Subject public key info	4.1.2.7	M		

algorithm		M		If any of the curves specified below for the `parameters` field is used, the following OID must be used, as specified in RFC 5480 and RFC 5639:
				`1.2.840.10045.2.1` (id-ecPublicKey)
				For curves Ed25519 or Ed448, one of the following OIDs must be used, as specified in RFC 8410:
				`1.3.101.112(Curve Ed25519)`
				`1.3.101.113(Curve Ed448)`

parameters		C		This field must only be present when the `algorithm` field contains the OID `1.2.840.10045.2.1`. Implicitly specify curve parameters through an OID associated with one of the following curves specified in NIST SP 800-186: `1.2.840.10045.3.1.7` (Curve P-256) `1.3.132.0.34` (Curve P-384) `1.3.132.0.35` (Curve P-521) Or one of the following curves specified in RFC 5639: `1.3.36.3.3.2.8.1.1.7` (brainpoolP256r1) `1.3.36.3.3.2.8.1.1.9` (brainpoolP320r1) `1.3.36.3.3.2.8.1.1.11` (brainpoolP384r1) `1.3.36.3.3.2.8.1.1.13` (brainpoolP512r1)
subjectPublicKey		M		For all curves except Ed25519 or Ed448, the public key shall be encoded in uncompressed form.
X.509v3 extensions	4.2	M		Further extensions may be present if they are marked non-critical.
Authority key identifier	4.2.1.1	M	NC	
keyIdentifier		M		Same value as the subject key identifier of the Issuer CA certificate

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Subject key identifier	4.2.1.2	M	NC	SHA-1 hash of the subjectPublicKey `BIT STRING` value (excluding tag, length, and number of unused bits).
Key usage	4.2.1.3	M	C	
Digital signature				`1` (mandatory)
Non-repudiation				`0`
Key encipherment				`0`
Data encipherment				`0`
Key agreement				`0`
Key certificate signature				`0`
CRL signature				`0`
Encipher only				`0`
Decipher only				`0`
Issuer alternative name	4.2.1.7	C	NC	Conditional, this extension shall be present if the certificate is issued by an IACA. The issuer alternative name extension shall provide contact information for the issuer of the certificate. For that purpose, the issuer alternative name shall include at least one of — `rfc822Name`, or — `uniformResourceIdentifier`. NOTE 2 This contact information is intended to help establish trust in the certificate and the certified key by appropriate out of band mechanisms. Note that this information is only meant for contact information and does not in itself imply any level of trust in the certificate.
Extended key usage	4.2.1.12	M	C	
		M		Mandatory: `1.0.18013.5.1.6` (mdlReaderAuth) Recommended: `1.0.23220.4.1.6` (mdocReaderAuth)
CRLDistributionPoints	4.2.1.13	M	NC	The ‘reasons’ and ‘cRL Issuer’ fields shall not be used. Editor’s note: Consider making this field optional for short-lived certificates
distributionPoint		M		URI for CRL distribution point
Private internet extensions				
Authority information access	4.2.2.1	C	NC	Conditional, shall be present if the CA issuing this certificate has an OCSP service or would like to indicate other Access Description elements.
Access description OCSP		C		Conditional, shall be present if the CA issuing this certificate has an OCSP service.
accessMethod		M		`1.3.6.1.5.5.7.48.1` (OCSP)
accessLocation		M		URI for corresponding OCSP service
Signature algorithm	4.1.1.2	M		Options: `1.2.840.10045.4.3.2` (ECDSA-with SHA256) `1.2.840.10045.4.3.3` (ECDSA-with SHA384) `1.2.840.10045.4.3.4` (ECDSA with SHA512)

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Signature value	4.1.1.3	M		Value according to the signature algorithm. By creating this signature, the CA certifies the binding between the public key material and the subject of the certificate, i.e. the mDL reader.

Key

Presence:

M	mandatory
O	optional
C	conditional

Criticality:

C	critical
NC	not critical

B.1.6 OCSP signer certificate

The OCSP signer certificate is used to sign OCSP messages. See Table B.8 for details.

Table B.8 — OCSP signer certificate

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Version	4.1.2.1	M		Shall be `v3`.
Serial number	4.1.2.2	M		Non-sequential positive, non-zero integer, shall contain at least 63 bits of output from a CSPRNG, should contain at least 71 bits of output from a CSPRNG, maximum 20 octets.
Signature	4.1.2.3	M		Value shall match the OID in the signature algorithm (below).
Issuer	4.1.2.4	M		Same exact binary value as the subject of IACA certificate
Validity	4.1.2.5	M		
notBefore		M		Date on which the certificate validity period begins.
notAfter		M		If the OCSP signer certificate supports the CRLDistributionPoints extension: Maximum of 457 days after “notBefore” date. If the OCSP signer certificate supports the Revocation Checking of an Authorized Responder extension: Maximum of 90 days after “notBefore” date.

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Subject	4.1.2.6	M		countryName is mandatory. The value shall be in upper case and contain the ISO 3166-1 alpha-2 code of the issuing country, exactly the same value as in the issuing country data element. The countryName shall be PrintableString. stateOrProvinceName is optional. If this element is present in the IACA root certificate, this element shall be present and hold the same value. The value shall exactly match the value of the data element "issuing_jurisdiction", if that element is present on the mDL. organizationName is optional. Its value is at the discretion of the IACA. commonName shall be present. Its value is at the discretion of the IACA. localityName is optional. Its value is at the discretion of the IACA. serialNumber is optional. If present, it shall be a PrintableString. Attributes that have a DirectoryString and for which the encoding is not listed above syntax shall be either PrintableString or UTF8String.
Subject public key info	4.1.2.7	M		
algorithm		M		1.2.840.10045.2.1 (Elliptic curve)
parameters		M		Implicitly specify curve parameters through an OID associated with one of the following curves specified in NIST SP 800-186: 1.2.840.10045.3.1.7 (Curve P-256) 1.3.132.0.34 (Curve P-384) 1.3.132.0.35 (Curve P-521) Or one of the following curves specified in RFC 5639: 1.3.36.3.3.2.8.1.1.7 (brainpoolP256r1) 1.3.36.3.3.2.8.1.1.11 (brainpoolP384r1) 1.3.36.3.3.2.8.1.1.13 (brainpoolP512r1)
subjectPublicKey		M		Public key shall be encoded in uncompressed form.
X.509v3 extensions	4.2	M		Further extensions may be present if they are marked non-critical.
Authority key identifier	4.2.1.1	M	NC	
keyIdentifier		M		Same value as the subject key identifier of the IACA root certificate
Subject key identifier	4.2.1.2	M	NC	SHA-1 hash of the subject public key BIT STRING value (excluding tag, length, and number of unused bits).
Key usage	4.2.1.3	M	C	
Digital signature				1
Non-repudiation				0
Key encipherment				0
Data encipherment				0
Key agreement				0
Key certificate signature				0

Certificate component	Section in RFC 5280	Presence	Criticality	Description
CRL signature				0
Encipher only				0
Decipher only				0
Subject alternative name	4.2.1.6	O	NC	
Issuer alternative name	4.2.1.7	M	NC	The issuer alternative name extension shall provide contact information for the issuer of the certificate. For that purpose, the issuer alternative name shall include at least one of — `rfc822Name`, or — `uniformResourceIdentifier`. NOTE This contact information is intended to help establish trust in the certificate and the certified key by appropriate out of band mechanisms. Note that this information is only meant for contact information and does not in itself imply any level of trust in the certificate.
Extended key usage	4.2.1.12	M	C	
id-kp-OCSPSigning		M		OCSP signing delegation, see RFC 6960.
CRLDistributionPoints	4.2.1.13	C	NC	Either this extension or the Revocation checking of an authorized responder extension shall be present. The 'reasons' and 'cRL Issuer' fields shall not be used.
distributionPoint		M		URI for CRL distribution point
Revocation checking of an authorized responder		C	NC	See RFC 6960, 4.2.2.2.1. Either this extension or the CRLDistributionPoints extension shall be present.
id-pkix-ocsp-nocheck				
Signature algorithm	4.1.1.2	M		Options: `1.2.840.10045.4.3.2` (ECDSA-with SHA256) `1.2.840.10045.4.3.3` (ECDSA-with SHA384) `1.2.840.10045.4.3.4` (ECDSA with SHA512)
Signature value	4.1.1.3	M		Value according to the signature algorithm. By creating this signature, the CA certifies the binding between the public key material and the subject of the certificate.

Key

Presence:

M	mandatory
O	optional
C	conditional

Criticality:

C	critical
NC	not critical

B.1.7 — MSO revocation list signer certificate

The MSO revocation list signer certificate is used to sign the status or identifier list as defined in 12.3.6. See Table B.9 for details.

Table B.9 — MSO revocation list signer certificate

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Version	4.1.2.1	M		Shall be `v3`.
Serial number	4.1.2.2	M		Non-sequential positive, non-zero integer, shall contain at least 63 bits of output from a CSPRNG, should contain at least 71 bits of output from a CSPRNG, maximum 20 octets.
Signature	4.1.2.3	M		Value shall match the OID in the signature algorithm (below).
Issuer	4.1.2.4	M		According to the Certification Authority issuing the MSO revocation list signer certificate
Validity	4.1.2.5	M		
notBefore		M		Date on which the certificate validity period begins.
notAfter		M		Maximum of 1187 days after “notBefore” date
Subject	4.1.2.6	M		According to the Certification Authority issuing the MSO revocation list signer certificate
Subject public key info	4.1.2.7	M		
algorithm		M		`1.2.840.10045.2.1` (Elliptic curve)
parameters		M		Implicitly specify curve parameters through an OID associated with one of the following curves specified in NIST SP 800-186: `1.2.840.10045.3.1.7` (Curve P-256) `1.3.132.0.34` (Curve P-384) `1.3.132.0.35` (Curve P-521) Or one of the following curves specified in RFC 5639: `1.3.36.3.3.2.8.1.1.7` (brainpoolP256r1) `1.3.36.3.3.2.8.1.1.9` (brainpoolP320r1) `1.3.36.3.3.2.8.1.1.11` (brainpoolP384r1) `1.3.36.3.3.2.8.1.1.13` (brainpoolP512r1)
subjectPublicKey		M		Public key shall be encoded in uncompressed form.
X.509v3 extensions	4.2	M		Further extensions may be present if they are marked non-critical.
Authority key identifier	4.2.1.1	M	NC	
keyIdentifier		M		Same value as the subject key identifier of the issuer’s certificate
Subject key identifier	4.2.1.2	M	NC	SHA-1 hash of the subject public key `BIT STRING` value (excluding tag, length, and number of unused bits).
Key usage	4.2.1.3	M	C	
Digital signature				`1`
Non-repudiation				`0`
Key encipherment				`0`
Data encipherment				`0`
Key agreement				`0`
Key certificate signature				`0`
CRL signature				`0`
Encipher only				`0`
Decipher only				`0`
Extended key usage	4.2.1.12	O	O or M	

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Key usage		O		`1.3.6.1.5.5.7.3.TBD` (oauthStatusSigning)
CRLDistributionPoints	4.2.1.13	O	NC	The 'reasons' and 'cRL Issuer' fields shall not be used.
distributionPoint		O		URI for CRL distribution point
Signature algorithm	4.1.1.2	M		Options: `1.2.840.10045.4.3.2` (ECDSA-with SHA256) `1.2.840.10045.4.3.3` (ECDSA-with SHA384) `1.2.840.10045.4.3.4` (ECDSA with SHA512)
Signature value	4.1.1.3	M		Value according to the signature algorithm. By creating this signature, the CA certifies the binding between the public key material and the subject of the certificate, i.e. the MSO revocation list signer.
Key Presence: M mandatory O optional C conditional Criticality: C critical NC not critical				

B.2 CRL profile

An IACA shall generate certificate revocation information in accordance with the Certificate Revocation List (CRL) format specified in Table B.10. This CRL shall be a full and complete CRL, i.e. list all unexpired certificates issued by the IACA that have been revoked for any reason. This CRL may contain revocation information for IACA issued certificates that are not standardized in this document. An IACA shall not use an indirect or delta CRL.

If no certificates have been revoked since the last CRL was issued, an IACA shall issue a new CRL at least every 90 days. An IACA may issue CRLs more frequently than every 90 days.

If a certificate is revoked, the IACA shall issue a new CRL indicating this revocation within 48 hours.

The CRL profile as defined in Table B.10 shall be used.

Table B.10 — CRL profile

CRL component	Section in RFC 5280	Presence	Criticality	Comments
Version	5.1.2.1	M		Shall be `v2`.
Signature	5.1.2.2	M		Value shall match the OID in the signature algorithm (below).
Issuer name	5.1.2.3	M		Same exact binary value as Subject of IACA certificate
This update	5.1.2.4	M		Issue date of this CRL
Next update	5.1.2.5	M		The next CRL will be issued no later than the next update date.
Revoked certificates	5.1.2.6	C		Conditional, shall not be present if there are no revoked certificates. If present, shall not be empty. Each CRL entry in the revoked certificates list shall contain the serial number of the revoked certificate and the revocation date. CRL entry extensions shall not be used.

CRL component	Section in RFC 5280	Presence	Criticality	Comments
CRL extensions	5.2			Further extensions shall not be present.
Authority key identifier	5.2.1	M	NC	
keyIdentifier		M		Same value as the subject key identifier of the IACA certificate.
CRL number	5.2.3	M	NC	Sequential CRL number, increased monotonically at each new CRL issued.
Signature algorithm	5.1.1.1	M		Options: `1.2.840.10045.4.3.2` (ECDSA-with SHA256) `1.2.840.10045.4.3.3` (ECDSA-with SHA384) `1.2.840.10045.4.3.4` (ECDSA with SHA512)
Signature value	5.1.1.2	M		Value according to signature algorithm.

Key
Presence:
M mandatory
C conditional
Criticality:
C critical
NC not critical

B.3 Certificate path and CRL validation examples

B.3.1 Certificate path validation example

This subclause gives an example on how the requirements from 9.3.3 apply to the certification path validation of an end-entity certificate issued by the IACA. This example does not include all steps from the path validation procedure as defined in RFC 5280. Those steps can be included when appropriate.

Use the following items from RFC 5280, section 6.1.1 as input values:

a) path length is 0;

b) current date/time;

d) Take from the IACA certificate:

   1. issuer name,

   2. public key algorithm,

   3. public key,

   4. public key parameters associated with the public key.

Initialize the following values from RFC 5280, section 6.1.2:

g) working_public_key_algorithm from IACA certificate;

h) working_public_key from IACA certificate;

i) working_public_key_parameters from IACA certificate;

j) working_issuer_name from IACA certificate;

k) Path length is 0.

Perform the following steps from RFC 5280, section 6.1.3:

a) verify step 1), 2), 3) and 4). If step 3) determines the revocation status by means of a CRL, B.3.2 gives an example on how the CRL validation can be performed.

The following steps are to be performed.

— Assign the certificate subjectPublicKey to working_public_key.

— Assign the subjectPublicKeyInfo parameters, i.e. the namedCurve, to the working_public_key_parameters variable.

— Assign the certificate subjectPublicKey algorithm to the working_public_key_algorithm variable.

— Verify the extended key usage extension in the target certificate contains the identifier for the certificate type as specified in Annex B.

— Process all critical extensions present in the certificate. Reject the certificate if it contains a critical extension that is not recognized or that contains information that cannot be processed.

The following steps are to be performed.

— Process any other recognized non-critical extensions present in the certificate. Reject the certificate if these extensions contain information that cannot be processed.

— Verify that the countryName element in the subject of the IACA certificate and countryName element in the target certificate are the same.

B.3.2 CRL validation example

This subclause gives an example on how the requirements from 12.7.3 apply to the CRL validation and certificate revocation checking of an end-entity certificate issued by the IACA.

Use the following items from RFC 5280, section 6.3.1 as input values:

a) certificate: certificate serial number and issuer name.

Initialize the following values from RFC 5280, section 6.3.2:

b) initialize the cert_status to the special value UNREVOKED.

Perform the following steps from RFC 5280, section 6.3.3:

a) obtain the latest available CRL of the IACA that issued the certificate. The latest available CRL may be a cached list, e.g. on the mDL reader;

b) verify that the CRL and the target certificate are issued by the same IACA;

g) validate the signature on the CRL using the public key from the IACA certificate;

j) search for the target certificate on the CRL. If an entry is found that matches the certificate issuer and serial number set the cert_status variable to UNSPECIFIED. If no entry is found, the cert_status variable stays UNREVOKED.

If step a), b) or g) is not successful, set the cert_status variable to UNDETERMINED and terminate the process.

Annex C
(informative)

Verified issuer certificate authority list (VICAL) provider

C.1 mDL VICAL provider policy and security requirements

C.1.1 General

C.1.1.1 Overview

The decentralized PKI trust model adopted by the mDL requires a mechanism to distribute and disseminate the set of certification authorities' certificates by issuing authorities. Furthermore, the lack of a global organization having oversight over the mDL ecosystem and willing to play an operational role (as is the case of ICAO for the electronic passport) limits the possibility of having a single central repository with all the IACA certificates and working as the reference trust anchor for all mDL participants.

In this context, a mechanism referred to as VICAL is hereby described whereby an entity (Provider) can compile, operate and provide such a trust anchor in the form of a service to mDL participants. As this service plays a critical role on the overall security and interoperability of the mDLs, a minimum set of security requirements are defined.

The VICAL Provider shall document the service it provides in a policy of technical and procedural controls. The policy shall follow the structure of this annex and be compliant with the respective requirements, which are mostly based on ISO/IEC 15408 and ISO/IEC 19790. VICAL providers can further extend the policy with additional matters (e.g. business, legal, etc.). This policy should not be the only item used to assess the trustfulness of a VICAL Provider.

This annex does not prescribe the nature or governance of a VICAL Provider. In particular, it does not preclude a scenario where multiple VICAL Providers may coexist, from public and/or private entities, competing and/or collaborating. However, it is expected that the VICAL information provided is consistent amongst different VICAL Providers.

Finally, the VICAL is provided as one possible mechanism of setting a secure and interoperable Trust Model. It does not preclude other possible mechanisms, such as bilateral and/or regional agreements.

This annex may be used for mDL as well as IDL as specified in ISO/IEC 18013-1, ISO/IEC 18013-2 and ISO/IEC 18013-3.

C.1.1.2 Document name and identification

This Policy is identified and can be referred to through the following OID:

```
id-idl-ml-policy OBJECT IDENTIFIER ::= {
    iso(1) standard(0) driving-licence (18013) part-5(5)
    VICAL(3) 1}
```

Policies of VICAL providers shall be uniquely identifiable, including all published revisions.

C.1.1.3 VICAL participants

This subclause provides an overview of the VICAL participants in the mDL.

C.1.1.3.1 VICAL providers

VICAL providers are organizations responsible for delivering the VICAL, including activities such as:

— finding and regularly validating issuing authorities' point of contacts;

— collecting information from issuing authorities through the respective point of contacts;

— compiling the VICAL;

— creating and securing the VICAL according to the defined format;

— distributing the VICAL amongst subscribers;

— updating the VICAL regularly.

C.1.1.3.2 Issuing authorities

For the definition of issuing authority, refer to Clause 3.

C.1.1.3.3 Issuing authorities' point of contact

Each issuing authority shall designate a point of contact, i.e. the unique endpoint for the trusted communication channel handling all communication between the VICAL Provider and the issuing authority. The point of contact may be the same as the initial contact point (see C.1.3.2).

The point of contact can have multiple formats, including but not limited to a person, P.O. box, an email address, dedicated telephone line, telefax, etc. See C.1.3.2 for limitations on the initial point of contact.

C.1.1.3.4 Subscribers

Subscribers receive the VICAL from the VICAL Provider and may redistribute it amongst relying parties, according to the licensing terms of the VICAL, if any.

A subscriber can simultaneously play the role of a relying party and vice-versa.

C.1.1.3.5 Relying parties

Relying parties make use of the received VICAL to validate the mDLs presented for validation.

C.1.1.4 VICAL usage

The VICAL contains the CA certificates intended to be used by relying parties as the trust anchor for the verification of authenticity and integrity of mDLs.

In the absence of VICALs that include all existing CA certificates from all Issuing Authorities it may happen that relying parties may have to use and combine several VICALs to increase coverage.

C.1.1.5 Policy administration

The VICAL Provider shall publish under this subclause the official contacts of the person or department responsible for the Policy.

The contacts shall include name, mailing address, telephone number, and email address as minimum information.

C.1.2 Publication and repository responsibilities

The VICAL Provider shall make the VICAL available to the subscribers in a secure, mutually trusted and authenticated channel, with periodic updates. The VICAL provider should make updates available at least every 90 days. The VICAL provider should also establish procedures for extraordinary updates for emergency cases.

C.1.3 Identification and authentication

C.1.3.1 Naming

Issuing Authorities are uniquely identified by the 2-letter ISO 3166-1 country code. In the special cases of multiple different Issuing Authorities within a country, the stateOrProvinceName / issuing jurisdiction (See Annex B and 7.2.1) can be used.

The naming of certification authorities used by an issuing authority shall follow the guidelines of the certificate profiles defined and the format established in VICAL syntax notation.

C.1.3.2 Initial identity validation

The initial identity validation of the issuing authority is a critical step on the overall security of the VICAL, as it is the first step to establishing a trusted communication channel between the VICAL provider and the issuing authorities.

After a trusted communication channel is established, the VICAL provider and the issuing authority can securely and reliably exchange information to be published in the VICAL.

As a minimum, the VICAL provider shall undertake the following steps.

a) Assess the legitimacy of and identify an initial contact point for the entity claiming to be an issuing authority. This can be achieved by a number of ways, including checking:

   — against a governmental authoritative source,

   — a regional/continental representative association (or similar body) of Issuing Authorities,

   — official recognized directorate of Issuing Authorities,

   — international conventions identifying Issuing Authorities,

   — official reference by another issuing authority previously trusted,

   — other methods that clearly and undoubtedly identify an issuing authority as legitimate.

b) Initiate contact with the issuing authority's initial contact point. Use this initial contact point to engage with the issuing authority in setting up a trusted communication channel and the protocol for its use. Sensitive material used to set up the trusted communication channel shall be exchanged using out-of-band communication. The protocol shall guarantee the integrity, authenticity, non-repudiation and confidentiality of the information to be exchanged.

NOTE An IACA link certificate is an example of a trusted communication channel.

After the trusted communication channel is established, the Initial Identity Validation is concluded and the parties can then start exchanging information required for the VICAL.

The VICAL provider shall regularly check that each assigned point of contact is still valid and the established communication channel provides the security guarantees, and perform the necessary updates if any required.

C.1.3.3 Identification and authentication for re-key requests

In the context of the VICAL, a re-key request is understood as the process to add a new CA to the records of the issuing authority.

This process shall use the trusted communication channel established at the initial identity validation (see C.1.3.2). If the trusted communication channel is not considered secure (for example, weak keys or algorithm in use, lost keys, key renewal, etc.), a new trusted communication channel shall be established following the same requirements.

C.1.3.4 Identification and authentication of suspension requests

Suspension requests from issuing authorities shall be communicated through the trusted communication channels established at the initial identity validation process (see C.1.3.2).

The VICAL Provider is also reserved the right to proceed unilaterally with the suspension of a CA on the grounds of a documented process and criteria.

C.1.4 VICAL life-cycle operational requirements

C.1.4.1 Certification authority application

The IA may apply to the VICAL Provider for inclusion of its CA into the VICAL.

The respective IA point of contact shall provide all requested information by the VICAL Provider, according to the established processes and through the trusted communication channel. The provided information shall be authentic, correct, complete and truthful.

The Terms and Conditions of the VICAL Service, if any, shall be made available to the applicant by the VICAL Provider.

NOTE An IACA link certificate is an example of a trusted communication channel.

C.1.4.2 Certification authority application processing

The VICAL Provider shall validate the information submitted by the IA's point of contact. Should any of the acceptance checks and conditions fail, the VICAL Provider is reserved the right to terminate the application process.

The VICAL Provider shall keep records (see C.1.5.5) of the application analysis, processing, internal and external checks and results.

C.1.4.3 Certification authority application acceptance

If the application processing by the VICAL Provider approves the CA, it can be included into the VICAL for the corresponding issuing authority and published according to the timeline set by the VICAL Provider and agreed by the applicant.

C.1.4.4 Certification authority renewal

A CA renewal is understood as a request to include a new CA certificate with the same name, public key, and other information as the old one, but with a new, extended validity period and a new serial number.

CA renewals shall not be used.

C.1.4.5 Certification authority application re-key

A CA re-key is understood as a request to include a new CA certificate with a different public key (and serial number) while retaining the remaining contents of the old CA certificate. The new certificate may be assigned a different validity period, key identifiers, specify a different CRL distribution point, and/or be signed with a different key.

The old certificate may or may not be revoked, but shall not be further re-keyed, renewed, or modified.

VICAL Providers shall process CA re-keys as new CA applications and thus follow the definitions laid down in C.1.4.1 to C.1.4.3.

C.1.4.6 Certification authority application modification

Modifying a CA certificate is understood as a request to include a new certificate that has the same or a different key and a different serial number, and that differs in one or more other fields from the old certificate.

The old certificate may or may not be revoked, but shall not be further re-keyed, renewed, or modified.

VICAL Providers shall process CA modifications as new CA applications and thus follow the definitions laid down in C.1.4.1 to C.1.4.3.

C.1.4.7 Certification authority application suspension

The VICAL Provider shall suspend CA certificates in a timely manner based on authorized and validated requests from the issuing authority point of contact.

The VICAL Provider shall keep records (see C.1.5.5) of the analysis, processing, internal and external checks and results.

C.1.4.8 End of subscription

This policy does not define nor limits any particular form or nature of relation between VICAL Providers and issuing authorities, and VICAL Providers and subscribers. However, it assumes the existence of these Participants (see C.1.1.3) and some form of relation is established between them.

In the event of termination of the relation between the VICAL Provider and the issuing authority, the VICAL Provider is allowed to remove/suspend or maintain unchanged the corresponding IACA on the VICAL, according to the terms and conditions agreed between the parties, or at its sole discretion in its absence.

On the other side, in the event of termination of the relation between the VICAL Provider and a subscriber, the contents of the VICAL shall not be affected.

C.1.5 Facility, management and operational controls

C.1.5.1 Physical security controls

VICAL Provider's equipment shall be protected from unauthorized access while the cryptographic module (see C.1.6.2) is installed and activated. The VICAL Provider shall implement physical access controls to reduce the risk of equipment tampering even when the cryptographic module is not installed and activated. VICAL Provider cryptographic tokens shall be protected against theft, loss, and unauthorized use.

The following controls shall be fulfilled.

a) Physical access to components of the VICAL Provider's system whose security is critical to the provision of its VICAL services shall be limited to authorized individuals.

b) Controls shall be implemented to avoid loss, damage or compromise of assets and interruption to business activities.

c) Controls shall be implemented to avoid compromise or theft of information and information processing facilities.

d) Components that are critical for the secure operation of the VICAL service shall be located in a protected security perimeter with physical protection against intrusion, controls on access through the security perimeter and alarms to detect intrusion.

e) The facilities concerned with VICAL generation and management (i.e. CAs status lifecycle management) shall be operated in an environment which physically protects the services from compromise through unauthorized access to systems or data.

f) Every entry to the physically secure area shall be subject to independent oversight and non-authorized person shall be accompanied by an authorized person whilst in the secure area.

g) Every entry and exit shall be logged and such access log shall be inspected periodically.

h) Physical protection shall be achieved through the creation of clearly defined security perimeters (i.e. physical barriers) around the VICAL generation and management services.

i) Any parts of the premises shared with other organizations shall be outside the perimeter of the VICAL generation management services.

j) Physical and environmental security controls shall be implemented to protect the facility housing system resources, the system resources themselves, and the facilities used to support their operation.

k) The VICAL Provider's physical and environmental security policy for systems concerned with VICAL generation and management services shall address the physical access control, natural disaster protection, fire safety factors, failure of supporting utilities (e.g. power, telecommunications), structure collapse, plumbing leaks, protection against theft, breaking and entering, and disaster recovery.

l) Controls shall be implemented to protect against equipment, information, media and software relating to the VICAL Provider's services being taken off-site without authorization.

m) Other functions relating to VICAL Provider's operations may be supported within the same secured area provided that the access is limited to authorized personnel.

C.1.5.2 Procedural controls

VICAL Providers shall implement security measures in order to protect the authenticity, integrity and confidentiality of their data and the accurate functionality of their IT systems.

The following controls shall be fulfilled.

a) The VICAL Provider shall administer user access of operators, administrators and system auditors.

b) The administration shall include user account management and timely modification or removal of access.

c) Access to information and application system functions shall be restricted in accordance with the access control policy.

d) The VICAL Provider's system shall provide sufficient computer security controls for the separation of trusted roles identified in VICAL Provider's practices, including the separation of security administration and operation functions. Particularly, use of system utility programs shall be restricted and controlled.

e) VICAL Provider's personnel shall be identified and authenticated before using critical applications related to the service.

f) VICAL Provider's personnel shall be accountable for their activities.

g) Activation of the VICAL signing key shall be under at least dual control by authorized, trusted personnel such that one person alone cannot activate the VICAL creation system on his/her own.

C.1.5.3 Personnel controls

The VICAL Provider shall ensure that employees and contractors support the trustworthiness of the VICAL Provider's operations.

The following controls shall be fulfilled.

a) The VICAL Provider shall employ staff and, if applicable, subcontractors, who possess the necessary expertise, reliability, experience, and qualifications and who have received training regarding security and personal data protection rules as appropriate for the offered services and the job function.

b) VICAL Provider's personnel should be able to fulfil the requirement of "expert knowledge, experience and qualifications" through formal training and credentials, or actual experience, or a combination of the two.

c) This should include regular (at least every 12 months) updates on new threats and current security practices.

d) Appropriate disciplinary sanctions shall be applied to personnel violating VICAL Provider's policies or procedures.

e) Security roles and responsibilities, as specified in the VICAL Provider's information security policy, shall be documented in job descriptions or in documents available to all concerned personnel.

f) Trusted roles, on which the security of the VICAL Provider's operation is dependent, shall be clearly identified.

g) Trusted roles shall be named by the management.

h) Trusted roles shall be accepted by the management and the person to fulfil the role.

i) VICAL Provider's personnel (both temporary and permanent) shall have job descriptions defined from the view point of roles fulfilled with segregation of duties and least privilege, determining position sensitivity based on the duties and access levels, background screening and employee training and awareness.

j) Where appropriate, job descriptions shall differentiate between general functions and VICAL Provider's specific functions. These should include skills and experience requirements.

k) Personnel shall exercise administrative and management procedures and processes that are in line with the VICAL Provider's information security management procedures.

l) Managerial personnel shall possess experience or training with respect to the VICAL service that is provided, familiarity with security procedures for personnel with security responsibilities and experience with information security and risk assessment sufficient to carry out management functions.

m) All VICAL Provider's personnel in trusted roles shall be free from conflict of interest that might prejudice the impartiality of the VICAL Provider's operations.

n) Trusted roles shall include roles that involve the following responsibilities:

   1. Security Officers: overall responsibility for administering the implementation of the security practices;

   2. System Administrators: authorized to install, configure, maintain and recover the VICAL Provider's trustworthy systems for service management;

3. System Operators: responsible for operating the VICAL Provider's trustworthy systems on a day-to-day basis. Authorized to perform system backup;

4. System Auditors: authorized to view archives and audit logs of the VICAL Provider's trustworthy systems.

o) VICAL Provider's personnel shall be formally appointed to trusted roles by senior management responsible for security requiring the principle of "least privilege" when accessing or when configuring access privileges.

p) Personnel shall not have access to the trusted functions until the necessary checks are completed.

C.1.5.4 Audit logging procedures

The VICAL Provider shall record and keep accessible for an appropriate period of time, including after the activities of the VICAL Provider have ceased, all relevant information concerning data issued and received by the VICAL Provider, in particular, for the purpose of providing evidence in legal proceedings and for the purpose of ensuring continuity of the service.

The following controls shall be fulfilled.

a) The confidentiality and integrity of current and archived records concerning operation of services shall be maintained.

b) Records concerning the operation of services shall be completely and confidentially archived in accordance with disclosed business practices.

c) Records concerning the operation of services shall be made available if required for the purposes of providing evidence of the correct operation of the services for the purpose of legal proceedings.

d) The precise time of significant VICAL Provider's environmental, key management and clock synchronization events shall be recorded.

e) The time used to record events as required in the audit log shall be synchronized with UTC at least once a day.

f) Records concerning services shall be held for a period of time as appropriate for providing necessary legal evidence and as notified in the VICAL Provider's terms and conditions.

g) The events shall be logged in a way that they cannot be easily deleted or destroyed (except if reliably transferred to long-term media) within the period of time that they are required to be held. Where possible, the security audit logs shall be automatically collected. Where this is not possible, a logbook, paper form, or other physical mechanism shall be used. All security audit logs, both electronic and non-electronic, shall be retained and made available during compliance audits.

C.1.5.4.1 Types of events recorded

The VICAL Provider shall record details of the actions taken to process a request and to issue a VICAL, including all information generated and documentation received in connection with the request; the time and date; and the personnel involved. The VICAL Provider shall make these records available to Auditors as proof of the VICAL Provider’s compliance with these requirements.

The VICAL Provider shall record at least the following events.

a) VICAL signing key lifecycle management events, including:

1. key generation, backup, storage, recovery, archival, and destruction; and

   2. cryptographic device lifecycle management events.

b) VICAL and IA lifecycle management events, including:

   1. CA application, re-key requests, and suspension;

   2. all verification activities stipulated in these requirements;

   3. date, time, phone number used, persons spoken to, and end results of verification telephone calls;

   4. acceptance and rejection of CA applications; and

   5. issuance of VICALs.

c) security events, including:

   1. successful and unsuccessful VICAL Provider's system access attempts;

   2. VICAL Provider's and security system actions performed;

   3. security profile changes;

   4. system crashes, hardware failures, and other anomalies;

   5. firewall and router activities; and

   6. entries to and exits from the VICAL Provider facility.

Log entries shall include the following elements:

d) date and time of entry;

e) identity of the person making the journal entry; and

f) description of the entry.

C.1.5.5 Records archival

The VICAL Provider shall retain the following for at least seven years after any CA (accepted or not) based on these records ceases to be valid:

a) log of all events relating to the life cycle of keys managed by the VICAL Provider's system;

b) documentation and other evidence as referred in C.1.4.2.

C.1.5.6 Key changeover

To minimize risk from compromise of a VICAL Provider's private signing key, that key may be changed often. From that time on, only the new key should be used to sign VICALs. If the old private key is necessary during a limited period of time to keep signing VICALs and allow the migration for Relying parties with legacy systems, the old key shall be retained and protected. Once the old private signing key is not needed anymore, it may be destroyed.

The VICAL Provider's signing key shall have a validity period as described in C.1.7.2.

When a VICAL Provider updates its private signature key and thus generates a new public key, the VICAL Provider shall notify all subscribers that rely on its respective VICALs that it has been changed. The VICAL Provider shall provide the new public key through secure means (e.g. trusted communication channel established with subscribers/relying parties, provision of key rollover certificates – new public key is signed by the old private key, and vice versa).

C.1.5.7 Compromise and disaster recovery

C.1.5.7.1 Incident and compromise handling procedures

System activities concerning access to IT systems, use of IT systems, and service requests shall be monitored.

The following controls shall be fulfilled.

a) Monitoring activities should take account of the sensitivity of any information collected or analysed.

b) Abnormal system activities that indicate a potential security violation, including intrusion into the VICAL Provider's network, shall be detected and reported as alarms.

c) The VICAL Provider shall monitor the following events:

   1. start-up and shutdown of the logging functions; and

   2. availability and utilization of needed services with the VICAL Provider's network.

d) The VICAL Provider shall act in a timely and co-ordinated manner in order to respond quickly to incidents and to limit the impact of breaches of security.

e) The VICAL Provider shall appoint trusted role personnel to follow up on alerts of potentially critical security events and ensure that relevant incidents are reported in line with the VICAL Provider's procedures.

f) The VICAL Provider shall establish procedures to notify the appropriate parties in line with the applicable regulatory rules of any breach of security or loss of integrity that has a significant impact on the VICAL service provided and on the personal data maintained therein within 24 hours of the breach being identified.

g) Where the breach of security or loss of integrity is likely to adversely affect a natural or legal person to whom the VICAL service has been provided, the VICAL Provider shall also notify the natural or legal person of the breach of security or loss of integrity without undue delay.

h) The VICAL Provider's systems shall be monitored including the monitoring or regular review of audit logs to identify evidence of malicious activity implementing automatic mechanisms to process the audit logs and alert personnel of possible critical security events.

i) The VICAL Provider shall address any critical vulnerability not previously addressed by the VICAL Provider, within a period of 48 hours after its discovery.

j) For any vulnerability, given the potential impact, the VICAL Provider may either choose to:

   1. create and implement a plan to mitigate the vulnerability; or

   2. document the factual basis for the VICAL Provider's determination that the vulnerability does not require remediation.

k) Incident reporting and response procedures shall be employed in such a way that damage from security incidents and malfunctions are minimized.

C.1.5.7.2 Computing resources, software, and/or data are corrupted

The following controls shall be fulfilled.

a) VICAL Provider's systems data necessary to resume CA operations shall be backed up and stored in safe places, preferably also remote, suitable to allow the VICAL Provider to timely go back to operations in case of incident/disasters.

b) Back-up copies of essential information and software should be taken regularly.

c) Adequate back-up facilities should be provided to ensure that all essential information and software can be recovered following a disaster or media failure.

d) Back-up arrangements should be regularly tested to ensure that they meet the requirements of business continuity plans.

e) Backup and restore functions shall be performed by the relevant trusted roles specified in C.1.5.3.

f) For information requiring dual control for management, for example keys, dual control shall be applied to recovery.

C.1.5.7.3 VICAL Provider private key compromise procedures

The following controls shall be fulfilled in case of a private key compromise:

a) The VICAL Provider's business continuity plan (or disaster recovery plan) shall address the compromise, loss or suspected compromise of a VICAL Provider's private key as a disaster.

b) The processes planned as per the previous requirement shall be in place.

c) Following a disaster, the VICAL Provider shall, where practical, take steps to avoid repetition of a disaster.

d) In the case of compromise as a minimum:

    1. the VICAL Provider shall inform the following of the compromise: all issuing authorities point of contacts and subscribers and other entities with which the VICAL Provider has agreements or other form of direct established relations, among which relying parties and VICAL Providers; and

    2. the VICAL Provider shall indicate that VICALs issued using this private key may no longer be valid.

Furthermore, the following controls shall be fulfilled in case of an algorithm compromise.

a) Should any of the algorithms, or associated parameters, used by the VICAL Provider or its issuing authorities point of contacts and/or subscribers become insufficient for its remaining intended usage, then the VICAL Provider shall inform all of them with whom it has agreement or other form of established relations.

b) Should any of the algorithms, or associated parameters, used by the VICAL Provider or its subscribers become insufficient for its remaining intended usage, then the VICAL Provider shall plan the transition to a new stronger algorithm and execute it the earliest possible time.

C.1.5.7.4 Business continuity capabilities after a disaster

The following controls shall be fulfilled.

a) The VICAL Provider shall define and maintain a continuity plan to enact in case of a disaster.

b) In the event of a disaster, including compromise of a private signing key or compromise of some other credential of the VICAL Provider, operations shall be restored within the delay established in the continuity plan, having addressed any cause for the disaster which may recur (e.g. a security vulnerability) with appropriate remediation measures.

C.1.5.8 VICAL termination

Potential disruptions to subscribers and relying parties shall be minimized as a result of the cessation of the VICAL Provider's services and, in particular, continued maintenance of information required to verify the correctness of VICAL services shall be provided.

Furthermore, the following controls shall be fulfilled.

a) The VICAL Provider shall have an up-to-date termination plan.

b) Before the VICAL Provider terminates its services, at least the following procedures apply.

    1. Before the VICAL Provider terminates its services, the VICAL Provider shall inform the following of the termination: all issuing authorities point of contacts and subscribers and other entities with which the VICAL Provider has agreements or other form of established relations, among which relying parties and VICAL Providers.

    2. Before the VICAL Provider terminates its services, the VICAL Provider shall terminate authorization of all subcontractors, if any, to act on behalf of the VICAL Provider in carrying out any functions relating to the processing and/or dissemination of the VICAL.

    3. Before the VICAL Provider terminates its services, the VICAL Provider shall transfer obligations to a reliable party for maintaining all information necessary to provide evidence of the operation of the VICAL Provider for a reasonable period, unless it can be demonstrated that the VICAL Provider does not hold any such information. The minimum information set is composed of:

        i. registration information;

        ii. event log archives.

    4. Before the VICAL Provider terminates its services, the VICAL Provider's private keys, including backup copies, shall be destroyed, or withdrawn from use, in a manner such that the private keys cannot be retrieved.

    5. Before the VICAL Provider terminates its services, where possible VICAL Provider should make arrangements to transfer provision of VICAL services for its existing subscribers and Issuing Authorities point of contacts to another VICAL Provider.

c) The VICAL Provider shall have an arrangement to cover the costs to fulfil these minimum requirements in case the VICAL Provider becomes bankrupt or for other reasons is unable to cover the costs by itself, as far as possible within the constraints of applicable legislation regarding bankruptcy.

d) The VICAL Provider shall state in its practices the provisions made for termination of service. This shall include:

1. notification of affected entities; and

2. where applicable, transferring the VICAL Provider's obligations to other parties.

e) The VICAL Provider shall maintain or transfer to a reliable party its obligations to make available its public key or its history of VICALs to subscribers and relying parties for a reasonable period.

C.1.6 Technical security controls

C.1.6.1 Key pair generation and installation

Appropriate security controls shall be in place for the management of any cryptographic keys and any cryptographic devices throughout their lifecycle.

The following controls shall be fulfilled.

a) The VICAL signing key pair generation shall be undertaken in a physically secured environment (see C.1.5.1) by personnel in trusted roles (see C.1.5.3).

b) The VICAL Provider key pair used for signing VICALs shall be created under, at least, dual control.

c) The number of personnel authorized to carry out VICAL signing key pair generation shall be kept to a minimum and be consistent with the VICAL Provider's practices.

d) VICAL Signing key pair generation shall be performed using an algorithm as specified in C.1.7.2.

e) The selected key length and algorithm for VICAL signing key are specified in C.1.7.2.

f) Before expiration of its VICAL signer certificate which is used for signing VICALs, in case of continuing with the service, the VICAL Provider shall generate a new key pair and obtain a corresponding VICAL signer certificate, and shall apply all necessary actions to avoid disruption to the operations of any entity that may rely on the VICAL signer certificate.

g) Before expiration of its VICAL signer certificate, in case of continuing with the service, the new VICAL signer certificate shall also be issued and distributed in accordance with this document.

h) The operations described in f) and g) above should be performed with a suitable interval between certificate expiry date and the last VICAL signed to allow all parties that have relationships with the VICAL Provider (subscribers, relying parties, Issuing Authorities, etc.) to be aware of this key changeover and to implement the required operations to avoid inconveniences and malfunctions. This does not apply to a VICAL Provider which will cease its operations before its own VICAL signer certificate expiration date."

i) The VICAL Provider shall have a documented procedure for conducting generation of VICAL signing key pairs. Such procedure shall indicate, at least, the following:

1. roles participating in the ceremony (internal and external from the organization);

2. functions to be performed by every role and in which phases;

3. responsibilities during and after the ceremony; and

4. requirements of evidence to be collected of the ceremony.

j) The VICAL Provider shall produce a report proving that the ceremony, as in i) above, was carried out in accordance with the stated procedure and that the integrity and confidentiality of the key pair was ensured.

k) CA signature verification (public) keys of the VICAL signing certificate shall be available to subscribers and relying parties in a manner that assures the integrity of the CA public key and authenticates its origin.

C.1.6.2 Private key protection and cryptographic module engineering controls

The following controls shall be fulfilled.

a) VICAL Provider's signing key pair generation shall be carried out within a secure cryptographic device which is a trustworthy system which:

   1. is assured to EAL 4 or higher in accordance with ISO/IEC 15408 or equivalent national or internationally recognized evaluation criteria for IT security provided this is a security target or protection profile which meets the requirements of this document, based on a risk analysis and taking into account physical and other non-technical security measures; or

   2. meets the requirements identified in ISO/IEC 19790 or FIPS 140-2 level 3.

b) The secure cryptographic device shall be operated in its configuration as described in the appropriate certification guidance documentation or in an equivalent configuration which achieves the same security objective.

c) The VICAL private signing key shall be held and used within a secure cryptographic device meeting the requirements item a) and b) above.

d) If and when outside the secure cryptographic device, the VICAL signing private key shall be protected in a way that ensures the same level of protection as provided by the secure cryptographic device.

e) The VICAL signing private key may be backed up, stored and recovered only by personnel in trusted roles (see C.1.5.3) using, at least, dual control in a physically secured environment (see C.1.5.1).

f) The number of personnel authorized to carry out the VICAL signing private key back up, storage and recovery shall be kept to a minimum and be consistent with the VICAL Provider's practices.

g) Copies of the VICAL private signing keys shall be subject to the same or greater level of security controls as keys currently in use.

h) Where the VICAL signing private keys and any copies are stored in a dedicated secure cryptographic device, access controls shall be in place to ensure that the keys are not accessible outside this device.

i) The secure cryptographic device shall not be tampered with during shipment.

j) The secure cryptographic device shall not be tampered with while stored.

k) The secure cryptographic device shall be functioning correctly.

l) The CA private signing keys stored on the CA's secure cryptographic device shall be destroyed upon device retirement.

C.1.6.3 Other aspects of key pair management

The VICAL Provider shall use appropriately the VICAL private signing keys.

The following controls shall be fulfilled.

a) The VICAL Provider shall not use the VICAL signing private keys beyond the end of their life cycle.

b) VICAL signing key(s) used for generating VICALs as defined in C.1.7.1 shall not be used for any other purpose.

c) The VICAL signing keys shall only be used within physically secure premises.

d) The use of the VICAL's private key shall be compatible with the hash algorithm, the signature algorithm and signature key length used for generating VICALs (defined in C.1.7.2).

e) All copies, if any, of the VICAL signing private keys shall be destroyed at the end of their life cycle.

C.1.6.4 Activation data

The installation, activation and recovery of the VICAL signing key pairs in a secure cryptographic device shall require simultaneous control of at least two trusted employees, for example, using m-of-n authentication mechanisms.

Data used to unlock private keys shall be protected from disclosure by a combination of cryptographic and physical access control mechanisms. Activation data shall be either:

— memorized; or

— biometric in nature; or

— recorded and secured at the level of assurance associated with the activation of the cryptographic module, and shall not be stored with the cryptographic module.

C.1.6.5 Computer security controls

The VICAL Provider's system access shall be limited to authorized individuals.

The following controls shall be fulfilled.

a) Controls (e.g. firewalls) shall protect the VICAL Provider's internal network domains from unauthorized access including access by subscribers and third parties.

b) Firewalls should also be configured to prevent all protocols and accesses not required for the operation of the VICAL Provider.

c) Sensitive data shall be protected against being revealed through re-used storage objects (e.g. deleted files) being accessible to unauthorized users.

d) Local network components (e.g. routers) shall be kept in a physically and logically secure environment.

e) Local network components (e.g. routers) configurations shall be periodically checked for compliance with the requirements specified by the VICAL Provider.

f) The VICAL Provider shall enforce multi-factor authentication for all accounts capable of directly causing VICAL issuance.

g) Dissemination application shall enforce access control on attempts to add or delete VICALs and modify other associated information.

h) Continuous monitoring and alarm facilities shall be provided to enable the VICAL Provider to detect, register and react in a timely manner upon any unauthorized and/or irregular attempts to access its resources.

C.1.6.6 Life cycle security controls

The VICAL Provider shall use trustworthy systems and products that are protected against modification and ensure the technical security and reliability of the processes supported by them.

The following controls shall be fulfilled.

a) An analysis of security requirements shall be carried out at the design and requirements specification stage of any systems development project undertaken by the VICAL Provider or on behalf of the VICAL Provider to ensure that security is built into IT systems.

b) Change control procedures shall be applied for releases, modifications and emergency software fixes of any operational software and changes to the configuration which applies the VICAL Provider's security policy.

c) The procedures shall include documentation of the changes.

d) The integrity of VICAL Provider's systems and information shall be protected against viruses, malicious and unauthorized software.

e) Media used within the VICAL Provider's systems shall be securely handled to protect media from damage, theft, unauthorized access and obsolescence.

f) Media management procedures shall protect against obsolescence and deterioration of media within the period of time that records are required to be retained.

g) Procedures shall be established and implemented for all trusted and administrative roles that impact on the provision of services.

h) The VICAL Provider shall specify and apply procedures for ensuring that:

   1. security patches are applied within a reasonable time after they come available;

   2. security patches are not applied if they introduce additional vulnerabilities or instabilities that outweigh the benefits of applying them; and

   3. the reasons for not applying any security patches are documented.

i) Capacity demands shall be monitored and projections of future capacity requirements shall be made to ensure that adequate processing power and storage are available.

C.1.6.7 Network security controls

The VICAL Provider shall protect its network and systems from attack.

The following controls shall be fulfilled.

a) The VICAL Provider shall segment its systems into networks or zones based on risk assessment considering functional, logical, and physical (including location) relationship between trustworthy systems and services.

b) The VICAL Provider shall apply the same security controls to all systems co-located in the same zone.

c) The VICAL Provider shall restrict access and communications between zones to those necessary for the operation of the VICAL Provider.

d) The VICAL Provider shall explicitly forbid or deactivate not needed connections and services.

e) The VICAL Provider shall review the established rule set on a regular basis.

f) The VICAL Provider shall keep all systems that are critical to the VICAL Provider's operation in one or more secured zone(s).

g) The VICAL Provider shall separate dedicated network for administration of IT systems and VICAL Provider 's operational network.

h) The VICAL Provider shall not use systems used for administration of the security policy implementation for other purposes.

i) The VICAL Provider shall separate the production systems for the VICAL Provider's services from systems used in development and testing (e.g. development, test and staging systems).

j) The VICAL Provider shall establish communication between distinct trustworthy systems only through trusted channels that are logically distinct from other communication channels and provide assured identification of its end points and protection of the channel data from modification or disclosure.

k) If a high level of availability of external access to the VICAL service is required, the external network connection should be redundant to ensure availability of the services in case of a single failure.

l) The VICAL Provider shall undergo or perform a regular vulnerability scan on public and private IP addresses identified by the VICAL Provider and record evidence that each vulnerability scan was performed by a person or entity with the skills, tools, proficiency, code of ethics, and independence necessary to provide a reliable report.

m) The VICAL Provider shall undergo a penetration test on the VICAL Provider's systems at set up and after infrastructure or application upgrades or modifications that the VICAL Provider determines are significant.

n) The VICAL Provider shall record evidence that each penetration test was performed by a person or entity with the skills, tools, proficiency, code of ethics, and independence necessary to provide a reliable report.

o) The VICAL Provider shall maintain and protect all VICAL systems in at least a secure zone and shall implement and configure a security procedure that protects systems and communications between systems inside secure zones and high security zones.

p) The VICAL Provider shall configure all VICAL systems by removing or disabling all accounts, applications, services, protocols, and ports that are not used in the VICAL's operations.

q) The VICAL Provider shall grant access to secure zones and high security zones to only trusted roles.

r) The VICAL issuing system shall be in a high security zone.

C.1.6.8 Timestamping

The following controls shall be fulfilled.

a) Asserted times shall be accurate to within three minutes.

b) Electronic or manual procedures may be used to maintain system time.

c) Clock adjustments are auditable events.

C.1.7 VICAL and VICAL signer certificate profiles

C.1.7.1 VICAL CDDL profile

The VICAL profile shall use the following CDDL structure:

```
VICAL = {
    "version" : Version             ; VICAL structure version, currently "1.0"
    "vicalProvider" : tstr          ; Identifies the VICAL provider
    ? "vicalIssueID" : uint         ; identifies the specific issue of the VICAL, shall be
unique and monotonically increasing
    "date" : tdate                  ; date-time of VICAL issuance
    ? "nextUpdate" : tdate          ; next VICAL is expected to be issued before this
date-time
    ? "notAfter": tdate             ; date after which the VICAL is not valid
"certificateInfos" : [*CertificateInfo]
    ? "extensions" : Extensions     ; Can be used for proprietary extensions
    ? "vicalURL": tstr              ; URL where this VICAL can be retrieved
    * tstr => RFU                   ; To be used for future extensions, all values are RFU
}

CertificateInfo = {
    "certificate" : bstr            ; DER-encoded X.509 certificate
    "serialNumber" : biguint        ; value of the  serial number field of the certificate
    "ski" : bstr                    ; value of the Subject Key Identifier field of the
certificate
    "docType" : [+ DocType]         ; DocType for which the certificate may be used as a
trust point
    ? "certificateProfile" : [+ CertificateProfile] ; Type of certificate
    ? "issuingAuthority" : tstr     ; Name of the certificate issuing authority
    ? "issuingCountry" : tstr       ; ISO3166-1 or ISO3166-2 depending on the issuing
authority
    ? "stateOrProvinceName" : tstr  ; State or province name of the certificate issuing
authority
    ? "issuer" : bstr               ; DER-encoded Issuer field of the certificate (i.e.
the complete Name structure)
    ? "subject" : bstr              ; DER-encoded Subject field of the certificate (i.e.
the complete Name structure)
    ? "notBefore" : tdate           ; value of the notBefore field of the certificate
    ? "notAfter" : tdate            ; value of the notAfter field of the certificate
    ? "extensions" : Extensions     ; Can be used for proprietary extensions
    * tstr => RFU                   ; To be used for future extensions, all values are RFU
}

Extensions = {* tstr => Ext}        ; Can be used for proprietary extensions

CertificateProfile = tstr           ; Uniform Resource Name (URN) according to RFC 8141
```

The CertificateInfo map contains the “docType” and “certificateProfile” key-value pairs. These pairs shall be used as follows.

— Each certificate in the VICAL is used by the associated issuing authority as a trust point to allow the verification of end-entity certificates in a given document ecosystem. The "docType" pair contains the document type(s) of the documents for which the current certificate is used. For the mDL ecosystem, the mDL doc type as specified in 13.4.1 shall be used. Relying parties should not attempt to use the certificate as a trust point for an ecosystem whose document type is not listed in the "docType" key-value pair.

— The way in which a certificate may be used depends not only on the ecosystem for which it is a trustpoint, but optionally also on the certificate profile. The intended use(s) of each certificate is indicated by the "certificateProfile" pair. For an mDL IACA root certificate, the mDL IACA OID specified in B.1.1 shall be used.

  EXAMPLE Within the mDL ecosystem the use of IACA root certificates is mandatory as the trust point for issuer data authentication. However, next to that optionally another type of (root) certificate may be used as the trust point for mdoc reader authentication. Within other document ecosystems, other security mechanisms may be used.

The `tdate` data types in the VICAL structure shall be encoded as specified in 7.1.

The `Extensions` structures may be used by the VICAL provider for priopprietary extensions.

The VICAL is encapsulated and signed by the untagged `COSE_Sign1` structure as defined in RFC 9052. Within the `COSE_Sign1` structure, the `payload` shall be `VICAL`. The `external_aad` field used in the `Sig_structure` shall be a bytestring of size zero.

The `alg` element (RFC 9052) shall be included as an element in the protected header. Other elements should not be present in the protected header.

The certificate containing the public key belonging to the private key used to sign the VICAL shall be included as an `x5chain` element as described in RFC 9360. It shall be included as an unprotected header element.

The VICAL provider should use one of the signature algorithms as defined in RFC 9053 for calculating the signature over the VICAL: "ES256", "ES384", "ES512" or "EdDSA". The VICAL provider should use one of the elliptic curves as specified in Table 22.

NOTE A VICAL can also be used to support documents that are not mdocs, for example a driving license compliant with ISO/IEC 18013-2 and ISO/IEC 18013-3. In such a case the doctype field can reflect the ISO/IEC 7816-4 AID of the application. In this example this would be "org.iso.7816.4.A0000002480200" for the standard encoding version of the IDL.

C.1.7.2 VICAL signer certificate profile

This certificate profile defines the VICAL signer certificates, establishing the minimum security parameters (key lengths, algorithms, policy IDs, etc.). CAs issuing certificates to VICAL Providers may choose to use equivalent or higher parameters, as well as other certificate fields and extensions that do not limit or reduce the overall security level. See Table C.1 for details.

This certificate uses an OID for the VICAL policy. The OID has the following definition:

id-mdl OBJECT IDENTIFIER ::= { iso(1) standard(0) 18013 5 }

id-mdl-kp OBJECT IDENTIFIER ::= { id-mdl 1 } - - arc for extended key purposes

id-mdl-kp-mdlVICAL OBJECT IDENTIFIER ::= { id-mdl-kp 8 } - - arc for mDL VICAL

Table C.1 — VICAL signer certificate profile

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Version	4.1.2.1	M		Shall be `v3`.
Serial number	4.1.2.2	M		Non-sequential positive, non-zero integer, shall contain at least 63 bits of output from a CSPRNG, should contain at least 71 bits of output from a CSPRNG, maximum 20 octets.
Signature	4.1.2.3	M		Value shall match the OID in the signature algorithm (below).
Issuer	4.1.2.4	M		According to the Certification Authority issuing the VICAL signer certificate
Validity	4.1.2.5	M		
notBefore		M		Date on which the certificate validity period begins.
notAfter		M		Maximum of 1 187 days after “notBefore” date
Subject	4.1.2.6	M		Minimum required fields. Others may be present (e.g. Serial Number, State, Organization Unit, etc.).
Country (C)		M		Country code of jurisdiction of VICAL Provider. Encoded as `PrintableString`.
Organization (O)		M		Full registered name of the VICAL Provider. Encoded as `UTF8String`.
Common name (CN)		M		Name under which VICAL Provider operates VICAL service and is commonly known. Encoded as `UTF8String`.
Subject public key info	4.1.2.7	M		
algorithm		M		`1.2.840.10045.2.1` (Elliptic curve)
parameters		M		Implicitly specify curve parameters through an OID associated with one of the following curves specified in NIST SP 800-186: `1.2.840.10045.3.1.7` (Curve P-256) `1.3.132.0.34` (Curve P-384) `1.3.132.0.35` (Curve P-521) Or one of the following curves specified in RFC 5639: `1.3.36.3.3.2.8.1.1.7` (brainpoolP256r1) `1.3.36.3.3.2.8.1.1.9` (brainpoolP320r1) `1.3.36.3.3.2.8.1.1.11` (brainpoolP384r1) `1.3.36.3.3.2.8.1.1.13` (brainpoolP512r1)
subjectPublicKey		M		Public key shall be encoded in uncompressed form.
X.509v3 extensions	4.2	M		Further extensions may be present if they are marked non-critical.
Authority key identifier	4.2.1.1	M	NC	
keyIdentifier		M		Same value as the subject key identifier of the issuer’s certificate
Subject key identifier	4.2.1.2	M	NC	SHA-1 hash of the subject public key `BIT STRING` value (excluding tag, length, and number of unused bits).
Key usage	4.2.1.3	M	C	
Digital signature				`0`
Non-repudiation				`1`
Key encipherment				`0`
Data encipherment				`0`
Key agreement				`0`

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Key certificate signature				`0`
CRL signature				`0`
Encipher only				`0`
Decipher only				`0`
Extended key usage	4.2.1.12	M	C	
Key usage		M		Mandatory: `1.0.18013.5.1.8` (mdlVICAL) Recommended: `1.0.23220.4.1.8`(mdocVICAL)
CRLDistributionPoints	4.2.1.13	M	NC	The 'reasons' and 'cRL Issuer' fields shall not be used.
distributionPoint		M		URI for CRL distribution point
Private internet extensions				
Authority information access	4.2.2.1	C	NC	Conditional, shall be present if the VICAL has an OCSP service.
Access description OCSP		M		
accessMethod		M		`1.3.6.1.5.5.7.48.1` (OCSP)
accessLocation		M		URI for corresponding OCSP service
Signature algorithm	4.1.1.2	M		Options: `1.2.840.10045.4.3.2` (ECDSA-with SHA256) `1.2.840.10045.4.3.3` (ECDSA-with SHA384) `1.2.840.10045.4.3.4` (ECDSA with SHA512)
Signature value	4.1.1.3	M		Value according to the signature algorithm. By creating this signature, the CA certifies the binding between the public key material and the subject of the certificate, i.e. the VICAL Provider.

Key
Presence:
M mandatory
O optional
C conditional
Criticality:
C critical
NC not critical

C.1.7.3 VICAL endpoint

A VICAL provider MAY provide an endpoint where the VICAL can be retrieved using the `vicalURI` element in the VICAL structure. If this element is present the endpoint shall provide this endpoint.

To obtain the VICAL through the endpoint the mdoc reader shall send an an HTTP GET request to the URI in the VICAL.

In the successful response, the VICAL provider shall use the content type "application/cwt". The content shall be the VICAL.

The HTTP response should use gzip content encoding as defined in RFC 9110.

A response may also choose to redirect the client to another URI using an HTTP status code in the 3xx range, which clients should follow. A client SHOULD detect and intervene in cyclical redirections (i.e., "infinite" redirection loops).

C.1.8 Vical validation procedure

The mdoc reader shall perform the following or functionally equivalent steps to verify that the received VICAL is authentic and valid.

1. Validate the certificate(s) included in the VICAL header according to 12.7.3.
2. Verify the digital signature of the COSE_Sign1 structure with the VICAL payload (see C.1.7.1) using the working_public_key, working_public_key_parameters, and working_public_key_algorithm from the certificate validation procedure of step 1.
3. Validate that the `nextUpdate` date is in the future. If the `nextUpdate` date is in the past, the mdoc reader should obtain the latest VICAL from the VICAL Provider.

C.1.9 Compliance audit and other assessment

It is beyond the scope of this Policy to establish an auditing scheme for VICAL Providers. Nevertheless, it is understood that VICAL Providers should be able to publicly demonstrate compliance to this Policy and the security requirements.

As a minimum, VICAL Providers shall conduct self-audits on a periodic basis (at least yearly) to assess compliance to this Policy.

An independent third-party assessment can be achieved by an VICAL Provider based on the following principles.

— Auditor qualification: VICAL Provider selects a competent certification body that meets the requirements of ISO/IEC 17021-1 or ISO/IEC 27006.

— Audit basis: the Audit is based on ISO/IEC 27001 and ISO/IEC 27002.

— Checking requirement realisation: the audit and control does not only check that procedural security controls are specified but also that they are adhered to in practice. This also includes the initial identity validation, the receipt of IACA applications and the suspension/removal procedure for IACAs.

— Iteration of audits and controls: audits and controls is performed at least every three years. The Auditing Body and the VICAL Provider carry out a review at least once a year by a team of one or more auditors to ensure on going compliance with this Policy.

— Being not conformant: in the event that an audit indicates that the VICAL Provider is not conformant to this Policy, or its certification becomes invalid or expires, the VICAL Provider notifies its point of contacts of Issuing Authorities and subscribers.

— Availability of audit results: the certificate of conformity can be made available to Issuing Authorities, subscribers, relying parties and other possible stakeholders.

A VICAL Provider should implement an Information Security Management System (ISMS) for its VICAL Service in accordance to ISO/IEC 27001. The ISMS is based on an ISMS policy of which its scope is defined by this Policy and, if applicable, the associated Practice Statement.

Annex D
(informative)

Data structure examples

D.1 General

This annex contains examples for different structures used in the document. Since CBOR results in binary structures, a diagnostic notation will be used together with the binary encoding, whenever CBOR examples are made in this annex. The diagnostic notation will indicate binary embedded cbor data as << >>.

D.2 Data elements

D.2.1 Driving privileges

CBOR data:

```
82a37576656869636c655f63617465676f72795f636f646561416a69737375655f64617465d903ec6a32303138
2d30382d30396b6578706972795f64617465d903ec6a323032342d31302d3230a37576656869636c655f636174
65676f72795f636f646561426a69737375655f64617465d903ec6a323031372d30322d32336b6578706972795f
64617465d903ec6a323032342d31302d3230
```
In diagnostic notation:

```
[
    {
        "vehicle_category_code": "A",
        "issue_date": 1004("2018-08-09"),
        "expiry_date": 1004("2024-10-20")
    },
    {
        "vehicle_category_code": "B",
        "issue_date": 1004("2017-02-23"),
        "expiry_date": 1004("2024-10-20")
    }
]
```

D.2.2 Age_over_nn

Table D.1 describes the different situations for the answers to an age_over_nn request given the presence of certain data elements on the mDL.

Table D.1 — Situations for answers to age_over_nn requests

	mDL holder actual age				
	19	**21**	**30**	**60**	**64**
Attestation statements present on the mDL	age_over_21 = FALSE age_over_60 = FALSE	age_over_21 = TRUE age_over_60 = FALSE	age_over_21 = TRUE age_over_60 = FALSE	age_over_21 = TRUE age_over_60 = TRUE	age_over_21 = TRUE age_over_60 = TRUE

Business question	Verifier request	Response (by mDL holder actual age)				
Age equal to or above 18?	age_over_18	No response	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE
Age equal to or above 19?	age_over_19	No response	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE
Age equal to or above 20?	age_over_20	No response	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE
Age equal to or above 21?	age_over_21	age_over_21 = FALSE	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE
Age equal to or above 25?	age_over_25	age_over_21 = FALSE	No response	No response	age_over_60 is TRUE	age_over_60 is TRUE
Age equal to or above 30?	age_over_30	age_over_21 = FALSE	No response	No response	age_over_60 is TRUE	age_over_60 is TRUE
Age equal to or above 50?	age_over_50	age_over_21 = FALSE	No response	No response	age_over_60 is TRUE	age_over_60 is TRUE
Age equal to or above 60?	age_over_60	age_over_60 = FALSE	age_over_60 = FALSE	age_over_60 = FALSE	age_over_60 is TRUE	age_over_60 is TRUE
Age equal to or above 63?	age_over_63	age_over_60 = FALSE	age_over_60 = FALSE	age_over_60 = FALSE	No response	No response
Age equal to or above 64?	age_over_64	age_over_60 = FALSE	age_over_60 = FALSE	age_over_60 = FALSE	No response	No response
Age equal to or above 65?	age_over_65	age_over_60 = FALSE	age_over_60 = FALSE	age_over_60 = FALSE	No response	No response
Age below 18?	age_over_18[a]	No response	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE
Age below 19?	age_over_19 [a]	No response	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE
Age below 20?	age_over_20 [a]	No response	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE
Age below 21?	age_over_21 [a]	age_over_21 = FALSE	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE	age_over_21 is TRUE
Age below 25?	age_over_25 [a]	age_over_21 = FALSE	No response	No response	age_over_60 is TRUE	age_over_60 is TRUE
Age below 30?	age_over_30 [a]	age_over_21 = FALSE	No response	No response	age_over_60 is TRUE	age_over_60 is TRUE
Age below 50?	age_over_50 [a]	age_over_21 = FALSE	No response	No response	age_over_60 is TRUE	age_over_60 is TRUE
Age below 60?	age_over_60 [a]	age_over_60 = FALSE	age_over_60 = FALSE	age_over_60 = FALSE	age_over_60 is TRUE	age_over_60 is TRUE
Age below 63?	age_over_63 [a]	age_over_60 = FALSE	age_over_60 = FALSE	age_over_60 = FALSE	No response	No response
Age below 64?	age_over_64 [a]	age_over_60 = FALSE	age_over_60 = FALSE	age_over_60 = FALSE	No response	No response
Age below 65?	age_over_65 [a]	age_over_60 = FALSE	age_over_60 = FALSE	age_over_60 = FALSE	No response	No response

[a] The business question “Is the mDL holder under the age of x” is converted into the question “Is the mDL holder of age x or older”, since this is the question the data structure is designed to answer. A response indicating that the mDL holder is of age x or older logically can be converted into the statement “the mDL holder is not under the age of x”. Likewise, a response

indicating that the mDL holder is not of age x or older can logically be converted into the statement “the mDL holder is under the age of x”.

D.3 Device engagement

D.3.1 Device engagement structure

This is an example of a device engagement structure for QR device engagement.

CBOR data:

```
a30063312e30018201d818584ba4010220012158205a88d182bce5f42efa59943f33359d2e8a968ff289d93e5f
a444b624343167fe225820b16e8cf858ddc7690407ba61d4c338237a8cfcf3de6aa672fc60a557aa32fc670281
830201a300f401f50b5045efef742b2c4837a9a3b0e1d05a6917
```

In diagnostic notation:

```
{
  0: "1.0",
  1:
  [
    1,
    24(<<
      {
        1: 2,
        -1: 1,
        -2: h'5A88D182BCE5F42EFA59943F33359D2E8A968FF289D93E5FA444B624343167FE',
        -3: h'B16E8CF858DDC7690407BA61D4C338237A8CFCF3DE6AA672FC60A557AA32FC67'
      }
    >>)
  ],
  2:
  [
    [
      2,
      1,
      {
        0: false,
        1: true,
        11: h'45EFEF742B2C4837A9A3B0E1D05A6917'
      }
    ]
  ]
}
```

D.3.2 NFC Handover request

A handover request example containing a request for NFC, BLE and Wi-Fi Aware transmission technologies:

```
91022548721591020263720102110204616301013000110206616301036e666300510204616301015700la201e
016170706c69636174696f6e2f766e642e626c7565746f6f74682e6c652e6f6f6230081b28078080bf2801021c
021107c832fff6d26fa0beb34dfcd555d4823a1c11010369736f2e6f72673a31383031333a6e66636e6663015a
172b016170706c69636174696f6e2f766e642e7766612e6e616e57030101032302001324fec9a70b97ac9684a4
e326176ef5b981c5e8533e5f00298cfccbc35e700a6b020414
```

D.3.3 NFC Handover select

A handover select example as a response to the D.3.2 example. The BLE transmission technology is selected:

```
91020f487315d10209616301013001046d646f631a200c016170706c69636174696f6e2f766e642e626c756574
6f6f74682e6c652e6f6f6230081b28128b37282801021c015c1e580469736f2e6f72673a31383031333a646576
696365656e676167656d656e746d646f63a20063312e30018201d818584ba4010220012158205a88d182bce5f4
2efa59943f33359d2e8a968ff289d93e5fa444b624343167fe225820b16e8cf858ddc7690407ba61d4c338237a
8cfcf3de6aa672fc60a557aa32fc67
```

D.4 Data retrieval

D.4.1 Device retrieval

D.4.1.1 mdoc request

CBOR data:

```
a26776657273696f6e63312e306b646f63526571756573747381a26c6974656d7352657175657374d8185893a2
67646f6354797065756f72672e69736f2e31383031332e352e312e6d444c6a6e616d65537061636573a1716f72
672e69736f2e31383031332e352e31a66b66616d696c795f6e616d65f56f646f63756d656e745f6e756d626572
f57264726976696e675f70726976696c65676573f56a697373756555f64617465f56b6578706972795f64617465
f568706f727472616974f46a726561646572417574688443a10126a118215901b7308201b330820158a0030201
0202147552715f6add323d4934a1ba175dc945755d8b50300a06082a8648ce3d04030230163114301206035504
030c0b72656164657220726f6f74301e170d3230313030313030303030305a170d32333132333130303030303030
5a3011310f300d06035504030c067265616465723059301306072a8648ce3d020106082a8648ce3d0301070342
0004f8912ee0f912b6be683ba2fa0121b2630e601b2b628dff3b44f6394eaa9abdbcc2149d29d6ff1a3e091135
177e5c3d9c57f3bf839761eed02c64dd82ae1d3bbfa38188308185301c0603551d1f041530133011a00fa00d82
0b6578616d706c652e636f6d301d0603551d0e04160414f2dfc4acafc5f30b464fada20bfcd533af5e07f5301f
0603551d23041830168014cfb7a881baea5f32b6fb91cc29590c50dfac416e300e0603551d0f0101ff04040302
078030150603551d250101ff040b3009060728818c5d050106300a06082a8648ce3d0403020349003046022100
fb9ea3b686fd7ea2f0234858ff8328b4efef6a1ef71ec4aae4e307206f9214930221009b94f0d739dfa84cca29
efed529dd4838acfd8b6bee212dc6320c46feb839a35f658401f3400069063c189138bdcd2f631427c58942411
3fc9ec26cebcacacfcdb9695d28e99953becabc4e30ab4efacc839a81f9159933d192527ee91b449bb7f80bf
```

In diagnostic notation:

```
{
  "version": "1.0",
  "docRequests":
  [
    {
      "itemsRequest":
      24(<<
        {
          "docType": "org.iso.18013.5.1.mDL",
          "nameSpaces":
          {
            "org.iso.18013.5.1":
            {
              "family_name": true,
              "document_number": true,
              "driving_privileges": true,
              "issue_date": true,
              "expiry_date": true,
              "portrait": false
            }
          }
        }
      >>),
      "readerAuth":
      [
        << {1: -7} >>
        {
          33:
h'308201B330820158A003020102021475527l5F6ADD323D4934A1BA175DC945755D8B50300A06082A8648CE3D
```

```
      "readerAuth":
      [
        << {1: -7} >>
        {
          33:
h'308201B330820158A00302010202147552715F6ADD323D4934A1BA175DC945755D8B50300A06082A8648CE3D
04030230163114301206035504030C0B72656164657220726F6F74301E170D3230313030313030303030305A17
0D3233313233313030303030305A3011310F300D06035504030C067265616465723059301306072A8648CE3D02
0106082A8648CE3D03010703420004F8912EE0F912B6BE683BA2FA0121B2630E601B2B628DFF3B44F6394EAA9A
BDBCC2149D29D6FF1A3E091135177E5C3D9C57F3BF839761EED02C64DD82AE1D3BBFA38188308185301C060355
1D1F041530133011A00FA00D820B6578616D706C652E636F6D301D0603551D0E04160414F2DFC4ACAFC5F30B46
4FADA20BFCD533AF5E07F5301F0603551D23041830168014CFB7A881BAEA5F32B6FB91CC29590C50DFAC416E30
0E0603551D0F0101FF04040302078030150603551D250101FF040B3009060728818C5D050106300A06082A8648
CE3D0403020349003046022100FB9EA3B686FD7EA2F0234858FF8328B4EFEF6A1EF71EC4AAE4E307206F921493
0221009B94F0D739DFA84CCA29EFED529DD4838ACFD8B6BEE212DC6320C46FEB839A35'
        },
        null,
h'1F3400069063C189138BDCD2F631427C589424113FC9EC26CEBCACACFCDB9695D28E99953BECABC4E30AB4EF
ACC839A81F9159933D192527EE91B449BB7F80BF'
      ]
    }
  ]
}
```

D.4.1.2 mdoc response

CBOR data:

```
a36776657273696f6e63312e3069646f63756d656e747381a367646f6354797065756f72672e69736f2e31383
031332e352e312e6d444c6c6973737565725369676e6564a26a6e616d65537061636573a1716f72672e69736
f2e31383031332e352e3186d8185863a4686469676573744944006672616e646f6d58208798645b20ea200e1
9ffabac92624bee6aec63aceedecfb1b80077d22bfc20e971656c656d656e744964656e7469666965726b666
16d696c795f6e616d656c656c656d656e7456616c756563446f65d818586ca4686469676573744944036672
16e646f6d5820b23f627e8999c706df0c0a4ed98ad74af988af619b4bb078b89058553f44615d71656c656d6
56e744964656e7469666965726a69737375655f646174656c656c656d656e7456616c7565d903ec6a3230313
92d31302d3230d818586da4686469676573744944046672616e646f6d5820c7ffa307e5de921e67ba5878094
787e8807ac8e7b5b3932d2ce80f00f3e9abaf71656c656d656e744964656e7469666965726b6578706972795
f646174656c656c656d656e7456616c7565d903ec6a323032342d31302d3230d818586da468646967657374
4944076672616e646f6d582026052a42e5880557a806c1459af3fb7eb505d3781566329d0b604b845b5f9e687
1656c656d656e744964656e7469666965726f646f63756d656e745f6e756d6265726c656c656d656e7456616
c75656931323334353637383 9d818590471a4686469676573744944086672616e646f6d5820d094dad764a2e
b9deb5210e9d899643efbd1d069cc311d3295516ca0b024412d71656c656d656e744964656e7469666965726
8706f7274726169746c656c656d656e7456616c7565590412ffd8ffe000104a46494600010101009000900000
0ffdb004300130d0e110e0c13110f11151413171d301f1d1a1a1d3a2a2c2330453d4947443d43414c566d5d4
c51685241435f82606871757b7c7b4a5c869085778f6d787b76ffdb0043011415151d191d381f1f38764f434
f767676767676767676767676767676767676767676767676767676767676767676767676767676767676767
67676767676767676ffc0001108001800640301220002110103110 1ffc4001b000003010003010000000000000000
0000000005060401020307ffc40032100001030303020502030900000000000000010203040005110612211331
41551617122410781a1163542527391b2c1f1ffc40015010101000000000000000000000000000000001ffc40
01a1101010100030100000000000000000000000001411121316 1ffda000c03010002110311003f00a5bbde22d
a2329c7d692bc7d0d03f52cfb0ff75e7a7ef3e7709723a1d0dae146ddfbb3c039ce07ad2bd47a7e32dbb8dd1
d52d6ef4b284f64a480067dfb51f87ffb95ff00eb9ff14d215de66af089ce44b7dbde9cb6890a2838eddf180
78f7add62d411ef4db9b10a65d6b95a147381ea0d495b933275fe6bba75c114104a8ba410413e983dff004f5
af5d34b4b4cde632d0bf1fd1592bdd91c6411f3934c2fa6af6b54975d106dcf4a65ae56e856001ebc03c7ce2
9dd9eef1ef10fc447dc9da76ad2aee93537a1ba7e4f70dd8eff0057c6dffb5e1a19854a83758e54528750946
ec6704850cd037bceb08b6d7d2cc76d3317fc7b5cc04fb6707269c5c6e0c5b60ae549242123b0e493f602a07
5559e359970d98db89525456b51c951c8afa13ea8e98e3c596836783d5c63f5a61a99fdb7290875db4be88ab
384bbbbbfc7183fdeaa633e8951db7da396dc48524fb1a8bd611a5aa2a2432f30ab420a7a6d3240c718cf031
fa9ef4c9ad550205aa02951df4a1d6c8421b015b769db8c9229837ea2be8b1b0d39d0eba9c51484efdb8c0ef
```

```
d8d258daf3c449699f2edbd4584e7af9c64e3f96b9beb28d4ac40931e6478c8e76a24a825449501d867d2b1d
cdebae99b9c752ae4ecd6dde4a179c1c1e460938f9149ef655e515c03919a289cb3dca278fb7bf177f4faa82
9dd8ce3f2ac9a7ecde490971fafd7dce15eed9b71c018c64fa514514b24e8e4f8c5c9b75c1e82579dc1233df
ec08238f6add62d391acc1c5256a79e706d52d431c7a0145140b9fd149eb3a60dc5e88cbbc2da092411e9dc7
1f39a7766b447b344e847dcac9dcb5abba8d145061d43a6fcf1e65cf15d0e90231d3dd9cfe62995c6dcc5ca1
2a2c904a15f71dd27d451453e09d1a21450961cbb3ea8a956433b781f1ce33dfed54f0e2b50a2b71d84ed6db
18028a28175f74fc6bda105c529a791c25c4f3c7a11f71586268f4a66b726e33de9ea6f1b52b181c760724e4
7b514520a5a28a283ffd9d81858ffa4686469676573744944096672616e646f6d58204599f81beaa2b20bd0f
fcc9aa03a6f985befab3f6beaffa41e6354cdb2ab2ce471656c656d656e744964656e746966696572264726
976696e675f70726976696c656765736c656c656d656e7456616c756582a37576656869636c655f636174656
76f72795f636f646561416a69737375655f64617465d903ec6a323031382d30382d30396b657870697279795f6
4617465d903ec6a323032342d31302d3230a37576656869636c655f63617465676f72795f636f646561426a6
9737375655f64617465d903ec6a323031372d30322d32336b6578706972795f64617465d903ec6a323032342
d31302d32306a697373756572417574688443a10126a118215901f3308201ef30820195a00302010202143c4
416eed784f3b413e48f56f075abfa6d87eb84300a06082a8648ce3d04030230233114301206035504030c0b7
5746f706961206961636131 0b3009060355040613025553301e170d3230313030313030303030305a170d323
131303031303030303030305a302131123010060355040 30c0975746f706961206473310b300906035504061 30
255533059301306072a8648ce3d020106082a8648ce3d03010703420004ace7ab7340e5d9648c5a72a9a6f56
745c7aad436a03a43efea77b5fa7b88f0197d57d8983e1b37d3a539f4d588365e38cbbf5b94d68c547b5bc87
31dcd2f146ba381a83081a5301e0603551d120417301581136578616d706c65406578616d706c652e636f6d3
01c0603551d1f041530133011a00fa00d820b6578616d706c652e636f6d301d0603551d0e0416041414e2901
7a6c35621ffc7a686b7b72db06cd12351301f0603551d2304183016801454fa2383a04c28e0d930792261c80
c4881d2c00b300e0603551d0f0101ff04040302078030150603551d250101ff040b3009060728818c5d05010
2300a06082a8648ce3d0403020348003045022100977 17ab9016740c8d7bcdaa494a62c053bbdecce1383c1a
ca72ad08dbc04cbb202203bad859c13a63c6d1ad67d814d43e2425caf90d422422c04a8ee0304c0d3a68d590
3a2d81859039da66776657273696f6e63312e306f646967657374416c676f726974686d675348412d3235366
c76616c756544696765737473a2716f72672e69736f2e31383031332e352e31ad00582075167333b47b6c2bf
b86eccc1f438cf57af055371ac55e1e359e20f254adcebf01582067e539d6139ebd131aef441b445645dd831
b2b375b390ca5ef6279b205ed45710258203394372ddb78053f36d5d869780e61eda313d44a392092ad8e052
7a2fbfe55ae0358202e35ad3c4e514bb67b1a9db51ce74e4cb9b7146e41ac52dac9ce86b8613db555045820e
a5c3304bb7c4a8dcb51c4c13b65264f845541341342093cca786e058fac2d59055820fae487f68b7a0e87a74
9774e56e9e1dc3a8ec7b77e490d21f0e1d3475661aa1d0658207d83e507ae77db815de4d803b88555d0511d8
94c897439f5774056416a1c7533075820f0549a145f1cf75cbeeffa881d4857dd438d627cf32174b1731c4c3
8e12ca936085820b68c8afcb2aaf7c581411d2877def155be2eb121a42bc9ba5b7312377e068f660958200b3
587d1dd0c2a07a35bfb120d99a0abfb5df56865bb7fa15cc8b56a66df6e0c0a5820c98a170cf36e11abb724e
98a75a5343dfa2b6ed3df2ecfbb8ef2ee55dd41c8810b5820b57dd036782f7b14c6a30faaaae6ccd5054ce88
bdfa51a016ba75eda1edea9480c5820651f8736b18480fe252a03224ea087b5d10ca5485146c67c74ac4ec31
12d4c3a746f72672e69736f2e31383031332e352e312e5553a4005820d80b83d25173c484c5640610ff1a31c
949c1d934bf4cf7f18d5223b15dd4f21c0158204d80e1e2e4fb246d97895427ce7000bb59bb24c8cd003ecf9
4bf35bbd2917e340258208b331f3b685bca372e85351a25c9484ab7afcdf0d2233105511f778d98c2f544035
820c343af1bd1690715439161aba73702c474abf992b20c9fb55c36a336ebe01a876d6465766963654b65794
96e666fa1696465766963654b6579a4010220012158209631 3d6c63e24e3372742bfdb1a33ba2c897dcd68ab
8c753e4fbd48dca6b7f9a2258201fb3269edd418857de1b39a4e4a44b92fa484caa722c228288f01d0c03a2c
3d667646f6354797065756f72672e69736f2e31383031332e352e312e6d444c6c76616c6964697479496e666
fa3667369676e6564c074323032302d31302d30315431333a33303a30325a6976616c696446726f6dc074323
032302d31302d30315431333a33303a30325a6a76616c6964556e74696cc074323032312d31302d303154313
33a33303a30325a584059e64205df1e2f708dd6db0847aed79fc7c0201d80fa55badcaf2e1bcf5902e1e5a62
e4832044b890ad85aa53f129134775d733754d7cb7a413766aeff13cb2e6c6465766963655369676e6564a26
a6e616d65537061636573d81841a06a64657669636541757468a1696465766963654d61638443a10105a0f65
820e99521a85ad7891b806a07f8b5388a332d92c189a7bf293ee1f543405ae6824d6673746174757300
```

In diagnostic notation:

```
{
  "version": "1.0",
  "documents":
  [
    {
      "docType": "org.iso.18013.5.1.mDL",
      "issuerSigned":
      {
        "nameSpaces":
        {
          "org.iso.18013.5.1":
          [
            24(<<
              {
                "digestID": 0,
                "random":
h'8798645B20EA200E19FFABAC92624BEE6AEC63ACEEDECFB1B80077D22BFC20E9',
                "elementIdentifier": "family_name",
                "elementValue": "Doe"
              }
            >>),
            24(<<
              {
                "digestID": 3,
                "random":
h'B23F627E8999C706DF0C0A4ED98AD74AF988AF619B4BB078B89058553F44615D',
                "elementIdentifier": "issue_date",
                "elementValue": 1004("2019-10-20")
              }
            >>),
            24(<<
              {
                "digestID": 4,
                "random":
h'C7FFA307E5DE921E67BA5878094787E8807AC8E7B5B3932D2CE80F00F3E9ABAF',
                "elementIdentifier": "expiry_date",
                "elementValue": 1004("2024-10-20")
              }
            >>),
            24(<<
              {
                "digestID": 7,
                "random":
h'26052A42E5880557A806C1459AF3FB7EB505D3781566329D0B604B845B5F9E68',
                "elementIdentifier": "document_number",
                "elementValue": "123456789"
              }
            >>),
            24(<<
              {
                "digestID": 8,
                "random":
h'D094DAD764A2EB9DEB5210E9D899643EFBD1D069CC311D3295516CA0B024412D',
                "elementIdentifier": "portrait",
                "elementValue":
h'FFD8FFE000104A46494600010101009000900000FFDB004300130D0E110E0C13110F11151413171D301F1D1A
1A1D3A2A2C2330453D4947443D43414C566D5D4C51685241435F82606871757B7C7B4A5C869085778F6D787B76
FFDB0043011415151D191D381F1F38764F434F7676767676767676767676767676767676767676767676767676
767676767676767676767676767676767676767676767676FFC00011080018006403012200021101031101FFC4
001B0000030100030100000000000000000000000005060401020307FFC400321000010303030205020309000000
000000010203040005110612211331141551617122410781A1163542527391B2C1F1FFC4001501010100000000
00000000000000000000000001FFC4001A110101010003010000000000000000000000014111213161FFDA000C03
010002110311003F00A5BBDE22DA2329C7D692BC7D0D03F52CFB0FF75E7A7EF3E7709723A1D0DAE146DDFBB3C0
39CE07AD2BD47A7E32DBB8DD1D52D6EF4B284F64A480067DFB51F87FFB95FF00EB9FF14D215DE66AF089CE44B7
DBDE9CB6890A2838EDDF18078F7ADD62D411EF4DB9B10A65D6B95A147381EA0D495B933275FE6BBA75C114104A
8BA410413E983DFF004F5AF5D34B4B4CDE632D0BF1FD1592BDD91C6411F3934C2FA6AF6B54975D106DCF4A65AE
56E856001EBC03C7CE29DD9EEF1EF10FC447DC9DA76AD2AEE93537A1BA7E4F70DD8EFF0057C6DFFB5E1A19854A
83758E54528750946EC6704850CD037BCEB08B6D7D2CC76D3317FC7B5CC04FB6707269C5C6E0C5B60AE5492421
23B0E493F602A075559E359970D98DB89525456B51C951C8AFA13EA8E98E3C596836783D5C63F5A61A99FDB729
0875DB4BE88AB384BBBBBFC7183FDEAA633E8951DB7DA396DC48524FB1A8BD611A5AA2A2432F30AB420A7A6D32
40C718CF031FA9EF4C9AD550205AA02951DF4A1D6C8421B015B769DB8C9229837EA2BE8B1B0D39D0EBA9C51484
EFDB8C0EFD8D258DAF3C449699F2EDBD4584E7AF9C64E3F96B9BEB28D4AC40931E6478C8E76A24A825449501D8
67D2B1DCDEBAE99B9C752AE4ECD6DDE4A179C1C1E460938F9149EF655E515C03919A289CB3DCA278FB7BF177F4
FAA829DD8CE3F2AC9A7ECDE490971FAFD7DCE15EED9B71C018C64FA514514B24E8E4F8C5C9B75C1E82579DC123
3DFEC08238F6ADD62D391ACC1C5256A79E706D52D431C7A0145140B9FD149EB3A60DC5E88CBBC2DA092411E9DC
71F39A7766B447B344E847DCAC9DCB5ABBA8D145061D43A6FCF1E65CF15D0E90231D3DD9CFE62995C6DCC5CA12
A2C904A15F71DD27D451453E09D1A21450961CBB3EA8A956433B781F1CE33DFED54F0E2B50A2B71D84ED6DB180
28A28175F74FC6BDA105C529A791C25C4F3C7A11F71586268F4A66B726E33DE9EA6F1B52B181C760724E47B514
520A5A28A283FFD9'
              }
            >>),
            24(<<
              {
                "digestID": 9,
                "random":
h'4599F81BEAA2B20BD0FFCC9AA03A6F985BEFAB3F6BEAFFA41E6354CDB2AB2CE4',
                "elementIdentifier": "driving_privileges",
                "elementValue":
                [
                  {
                    "vehicle_category_code": "A",
                    "issue_date": 1004("2018-08-09"),
                    "expiry_date": 1004("2024-10-20")
                  },
                  {
                    "vehicle_category_code": "B",
                    "issue_date": 1004("2017-02-23"),
                    "expiry_date": 1004("2024-10-20")
                  }
                ]
              }
            >>)
          ]
        },
        "issuerAuth":
        [
          << {1: -7} >>,
          {
            33:
h'308201EF30820195A00302010202143C4416EED784F3B413E48F56F075ABFA6D87EB84300A06082A8648CE3D
0403023023311430120603550403C0B75746F706961206961636131 0B3009060355040613025553301E170D32
303130303130303030305A170D32313130303130303030305A302131123010060355040 3C0975746F7069
61206473310B3009060355040613025553305930130 6072A8648CE3D020106082A8648CE3D03010703420004AC
E7AB7340E5D9648C5A72A9A6F56745C7AAD436A03A43EFEA77B5FA7B88F0197D57D8983E1B37D3A539F4D58836
5E38CBBF5B94D68C547B5BC8731DCD2F146BA381A83081A5301E0603551D120417301581136578616D706C6540
6578616D706C652E636F6D301C0603551D1F041530133011A00FA00D820B6578616D706C652E636F6D301D0603
551D0E0416041414E29017A6C35621FFC7A686B7B72DB06CD12351301F0603551D2304183016801454FA2383A0
4C28E0D930792261C80C4881D2C00B300E0603551D0F0101FF040403020780301506 03551D250101FF040B3009
060728818C5D050102300A06082A8648CE3D0403020348003045022100977 17AB9016740C8D7BCDAA494A62C05
3BBDECCE1383C1ACA72AD08DBC04CBB202203BAD859C13A63C6D1AD67D814D43E2425CAF90D422422C04A8EE03
04C0D3A68D'
          },
          <<
            24(<<
              {
                "version": "1.0",
                "digestAlgorithm": "SHA-256",
                "valueDigests":
                {
                  "org.iso.18013.5.1":
                  {
                    0:
h'75167333B47B6C2BFB86ECCC1F438CF57AF055371AC55E1E359E20F254ADCEBF',
                    1:
h'67E539D6139EBD131AEF441B445645DD831B2B375B390CA5EF6279B205ED4571',
                    2:
h'3394372DDB78053F36D5D869780E61EDA313D44A392092AD8E0527A2FBFE55AE',
                    3:
h'2E35AD3C4E514BB67B1A9DB51CE74E4CB9B7146E41AC52DAC9CE86B8613DB555',
                    4:
h'EA5C3304BB7C4A8DCB51C4C13B65264F845541341342093CCA786E058FAC2D59',
                    5:
h'FAE487F68B7A0E87A749774E56E9E1DC3A8EC7B77E490D21F0E1D3475661AA1D',
                    6:
h'7D83E507AE77DB815DE4D803B88555D0511D894C897439F5774056416A1C7533',
                    7:
h'F0549A145F1CF75CBEEFFA881D4857DD438D627CF32174B1731C4C38E12CA936',
                    8:
h'B68C8AFCB2AAF7C581411D2877DEF155BE2EB121A42BC9BA5B7312377E068F66',
                    9:
h'0B3587D1DD0C2A07A35BFB120D99A0ABFB5DF56865BB7FA15CC8B56A66DF6E0C',
                    10:
h'C98A170CF36E11ABB724E98A75A5343DFA2B6ED3DF2ECFBB8EF2EE55DD41C881',
                    11:
h'B57DD036782F7B14C6A30FAAAAE6CCD5054CE88BDFA51A016BA75EDA1EDEA948',
                    12:
h'651F8736B18480FE252A03224EA087B5D10CA5485146C67C74AC4EC3112D4C3A'
                  },
                  "org.iso.18013.5.1.US":
                  {
                    0:
h'D80B83D25173C484C5640610FF1A31C949C1D934BF4CF7F18D5223B15DD4F21C',
                    1:
h'4D80E1E2E4FB246D97895427CE7000BB59BB24C8CD003ECF94BF35BBD2917E34',
                    2:
h'8B331F3B685BCA372E85351A25C9484AB7AFCDF0D2233105511F778D98C2F544',
                    3: h'C343AF1BD1690715439161ABA73702C474ABF992B20C9FB55C36A336EBE01A87'
                  }
                },
                "deviceKeyInfo":
                {
                  "deviceKey":
                  {
                    1: 2,
                    -1: 1,
                    -2:
h'96313D6C63E24E3372742BFDB1A33BA2C897DCD68AB8C753E4FBD48DCA6B7F9A',
                    -3:
h'1FB3269EDD418857DE1B39A4E4A44B92FA484CAA722C228288F01D0C03A2C3D6'
                  }
                },
                "docType": "org.iso.18013.5.1.mDL",
                "validityInfo":
                {
                  "signed": 0("2020-10-01T13:30:02Z"),
                  "validFrom": 0("2020-10-01T13:30:02Z"),
                  "validUntil": 0("2021-10-01T13:30:02Z")
                }
              }
            >>)
          >>,

h'59E64205DF1E2F708DD6DB0847AED79FC7C0201D80FA55BADCAF2E1BCF5902E1E5A62E4832044B890AD85AA5
3F129134775D733754D7CB7A413766AEFF13CB2E'
        ]
      },
      "deviceSigned":
      {
        "nameSpaces": 24(<< {} >>),
        "deviceAuth":
        {
          "deviceMac":
          [
            << {1: 5} >>,
            {},
            null,
            h' E99521A85AD7891B806A07F8B5388A332D92C189A7BF293EE1F543405AE6824D'
          ]
        }
      }
    }
  ],
  "status": 0
}
```

D.5 Security mechanisms

D.5.1 Session encryption

Taking the examples from D.3.2, D.3.3, D.4.1.1 and D.4.1.2 and the following key data:

Ephemeral device key:

```
  x: 5a88d182bce5f42efa59943f33359d2e8a968ff289d93e5fa444b624343167fe
  y: b16e8cf858ddc7690407ba61d4c338237a8cfcf3de6aa672fc60a557aa32fc67
  d: c1917a1579949a042f1ba9fc53a2df9b1bc47adf31c10f813ed75702d1c1f136
```

Ephemeral reader key:

```
  x: 60e3392385041f51403051f2415531cb56dd3f999c71687013aac6768bc8187e
  y: e58deb8fdbe907f7dd5368245551a34796f7d2215c440c339bb0f7b67beccdfa
  d: de3b4b9e5f72dd9b58406ae3091434da48a6f9fd010d88fcb0958e2cebec947c
```

CBOR data:

SessionEstablishment:

```
a26a655265616465724b6579d818584ba40102200121582060e3392385041f51403051f2415531cb56dd3f999c
71687013aac6768bc8187e225820e58deb8fdbe907f7dd5368245551a34796f7d2215c440c339bb0f7b67beccd
fa6464617461590 2df52ada2acbeb6c390f2ca0bc659b484678eb94dd45074386aadece23777b44606e42e2846
bc2e2ee3c1e867b1d1685e41354a021abb0fda36f09cf5d5c51b561d3be41c9347ae71cf2b49de9dec7b44046a
b02247931b210c9157840c1514a6027b08810716adf6196634497931 4ac3ae9f40e66e015c1254a684108bd093
e8772ec333fb663fd6803af02ea10bdbe83a999f75b55a180f872139fb57ac04acd58ca15eca150cde1c3b8494
01188b7a30ce887dd7b71b12eda2fc6ec6e5235a6c9498351fcd301f2292a4ebba7555285cee84ead96ef1677b
0af8239f6a7a52af4b8809b1d52ab21a162ca31ade21c57bd1d9970a2832aac41c7d52d1c4fee4ee64030a218d
f51363be701792fa6c515c489bd39dcad6fba48f1d6eb19e9c769531a3bf9998a32c01841305f23844ca3db6a1
ff0d0d917343d62fc72ad58eab01a3198116f19606609f94e35eacb78d23c59c67852a361915fe87848cdba563
0c99fab71aeff72d131cf442654f7708ec48216416f2d996cf6cf91012b771b88907b1d1629dfa794343e653c3
1207482e2f6621cd4b5dcf3b3c328625c33fe98be99c5f264a264315be41bafdc726f8bcde5920de0a71884d86
0af44c1ff1b3d78b2e8d720d85dae53fea2b3fa1806162a4be02d039567c5eb2419c2ad879af48fcb7df55ca94
f1b00f62187fa2329c8227aae0130ec052ca3e2102e57e72911b328cfdcfbaaf6b9364660f613415382644c30c
0bd4e222c5cf94ba5a73679c53d5ced95ca50787c2289a0c17358393c1e0f2272361002fb9b160606888a59ef7
a2c389f68b7cb424572db026b17cf2bdcafcb67c8292d92b50050356900a62a82b16f854759052b00f0f4673a4
6229f43257e8e8325401b3fecc8c6d2258baf7f7c2fbbafab3a1b6aded4eceac1eafd5b61118df93bc0a622b03
504fde47cebb224e983db12677e316c22aae042d6ce4adae0d8b0f40437b8e1afa0859c9501beb63974496859a
60f11069b1965b4ffac5779a96191f89eac7caa688b9e67c
```

SessionData:

```
a16464617461590dfa46da5fb292a7880af38f2e4d9eb23ecac231ad21dfe81e3b5c21e7f3d2b5a2d4676dd133
31112f0fba678275dc2fcd889150a7bbb333b7ecf5f35fc8b2e4aee701651a4bdc93cd25bc533582647507b5b9
a075deaf7e1ef035acb3c8b403ac6e51a19d4289381035199da169b5ab175d8bc2075ac73dbaa76aa79d9a26ac
393003451552 5c413110abaeae731545b36400205f3130e902242db99066a04ab6cef9d14672c3dfe467802e53
64dff5535c8c36fdb53afde285ee9462f72a4f8b5707879590d8b5ee83a3068c1c25f13681085cf4a5af3cb2e7
7bcc7cae6def76ab5ad119e6db563799d15af9f5861b7ec0003c68fa46f12ca366263b5fe5bb8f5b16bede1e5e
5919abf8b675fb10ce4655815fe6a3582ec44e0c93f0cf3a5ea1ca2e476113b47bc1ff2484f791c68385cd5ff3
f1f27f70a88c7c649581e59a5bb2371d6268704526bf1b16cd36d9e739bef50a199deafb8ceadd42c260e58688
bd569b420f32ad0502e6dff9346440459febbb49d843e50e93d46c0fd2125d4131e1d528435110b5e0db9e4179
5716422dc895425773eae490fbfba7c10f85ec364bdfac7de120ba4883142e854bf19510f969be690fad9a3f71
97885dcb44bd9028998adf6e95356af058e4c20502b5c4d3c4c52469727042e75e0ca8e43efb590da9a1a1cc3f
fdd03a422e7589ec237c36c0d5587ab853ab39be4388cdd9feb7b763ecb6344172fafde86a9501975dd86f19f0
95f98a1e65ace933e8723db8f1e5074f7c9c0415e4e69f12c2a6c14c0ca09c872f2ac83e0ec7294e6dc4f1fa08
7d48201c9c68d1cef4b793fb97b374505b348e090207a17930534759906 0200a1f398e9d9aa2eadbe31ee0172b
25d1f0b1287b3aad8f1afa560981c6490a1aec1052f47201ab5baed0ecf826c7b2b416ce0c2e265403a922f80d
1cf6842a7182c8bbe7a138096cb1fcffcfc102d16d08cbf86108393ef8276120e9341c11c5e5e1cebc257b697d
2182ed1ead67d7e5814a3f9b380cf917ccf59921f4ae545a14a8ec0f8642b7bd5565039b60546efb2c68aa0ac2
d3f972386cd9f47280e2f27af727335cd6ca1aec25c2ad2d079cd3fb045ccbaf6f4a5b0a80e07795f5790ff53a
aefdf3a63240ca7e9d5658ce84a21d310463335cf1f6015dc89ee54d687abcd4787c060a0bce53020b1c6a5bb6
c3ea4d759b7f3eddabcaade785c23caa9ba15272724e0589cef502d80085bd77bae93897b01e2588ddd504aebe
8333a153790a9699748a4269f5017c81aaf8061f9985e0cf7553b4c4d0a5c401f20f0ab6d85078728676bcfdc9
c1a1bc4dea2b01c68e0bb7bf8ada66aa93fc322673efa3895b09d085a0cd085dab3018a9e032acd3d715bf190c
629fc55d55ed815b0ba56e83f78c453165c024ca5b2de5b4accfbab07aa1c482b8e960db03652167f903db0ecf
03ef85834951a603c3e6069210744d70c5875e50badb91dcc8beca4945b869e73e535a7967fbcb377ce861f07e
537349c8a33793c0c05c1b4dd220634e1f52f6701d79d83f99311d6c1463ea3bbcf4d1a7614cbedfceafa1e874
b84e6b30698551c6da9f8487fc5270cd5dc73f9aa9dc8a513b1514d7abb0bec4d2cd06b14efd23f3c59602fcfc
5d9aecf48148128ca86d6d4ebbe2f68460db00a49759887a478a1eccdd8b426aa130d8b7da0e331415a9682d71
25e56285a927f54ac5139a3f8d78c2dc6ffa0bc9ad4db749e8dae10169aa6be9206b635943b7a0970a23e04a2b
a66712365b2ea0afb0422a0224592074f723781b8df86627296f9f126f94efe089f8db5f2eea4f28673e11ccd8
0726901ee1bcc8fa49b6f65c0f8587222f2bb81744a3ece2f74ac21e8c5805d054f93e2d669616ecbc07f3d017
a36b951aecd28a14ada87e4f935ef7f93c2c7d01cedf5658131908e4d36aa4690ba952d3f7fa25b6a8481d42f6
a7794c8b1dd4de2f7e6bb3ed6fd3915591997e06f01715bac3cdfb0885ce97136e4b7b4b1e6685fd42de1e8ae6
10523c2b2603977cf3c45058802d734322a9aa0181ff1231b4674071e5c75d8f11e56d67916b4d5f7a9b42a265
19c990fdb728630f0755cf8fc0d177c24e1375cc793cee463da5dbef1fe6a835bf75317052b7b077d397c7b617
a632371d22eb64c91f91d8fdf5aa3345f0e6e7a0d42e8da52087a89428a1241b4527d5f4d3751b10d40e8768bc
6b1a55b531a9e8fb9afdfaf3aed0a3bc71b9c432b89da34e4685b8b48f650394e760270df2c837ab51ded999b9
03c0f9630478adb8843e462ad8123466ff837dc18241db341ecabbc0a8693bd35831d49c09adf8c5d7c4f80b36
```

```
28b633d46a93c2b7b61e5127abb5e877e6904de7049a53213b357cdb443c31bababdc0480fe6986951ea9d684e
53f5b53871e80148afc30c4e2eed05a0aeafa0b98f98e8ffbad6af968ecfd286b24c43bf2b54e64e28fab328d7
91bbf80292e2226dfc7d0d562b511321b6b501eadbd1661f028bad02e37575104f646222484aac1548a762cfec
36b9597a4e7309497b49f1921a948a597cc7d721f6396389aaffd2828da0443e4f2354cf2fbe69c0a2a8f3d227
1207440e95445a4fb24de95d5232e66ff84c8e1ab5c5576e316ce3e78d7c0047695ef0037cc7f7d0f21fa53a19
76a7e1bb2d70751c6ebebb5fd24dad9fa58ffee261b0ee7a55a86a6236901feb32240096531317 5a263e5b8437
eb2a7ddf239d15c27b2e8b55abccd7574df807f3e8483670aac580fec1bbeb779a3cb2e89440e8badccdcdccf6
f6e7dd9a2d62080c7057824ede7186f924e2c59934a0fc0d518d05ff48da1b012322bb55d2493a850954b6af58
5e72f5bef55180f9fc293e40bd48a261cee4099ad4674541ddb5a81683034e9e31f59c8942321f09a5ed1e1185
4b6cd8af4afcf919f2001722c2396b9b30e1c5d8182061bb39201477a0538db896a399493151b1f6162284284b
8cc42701fad81c63846f0d7e3be5004a3d1e5dc047c2b28d7c28940e70ba52ac7aa0de584a9ff7859aa0e9af8d
24d5dcfd07cf5c6fd8034e2a3ef63b41be59ab93d941cee00b3029478181e576d3b0c09f28b1c70ab7c2e622e7
cacb2f6d2337c430afa5736169bddfe379f5b3d3392cfaa1bc45f6d3e47e73678ec19cecb20b3aee82378e41f8
4b59c172abc8d40370f8bb833f8edca890cd4f85f5610d3e208bada4deb6f15d9cb0a9f805b1f9915e40bed12e
d2494886801d6ff83df037b4a245d03489fa7dda552dccbd11b84101c4147595e0ee6d89b5d7621492d21dc01b
5afc1ce3c65cc878d823810f7d0c4232374a4e82013ad3a99f2422aa8e4e2cc33c66dc5b07bd2ece416999563a
6f2e62ef367904990f1442519eb00c6eecb1c886bde5613dba37908cc79f71bcfbf5fb96142488f0be69c510e1
44bd1a25f2e6f22bf78cc213d7ba83def9146224b67c1ca5cad8520e55d1b01daaa70475caa1e4ed117ac8952a
3f4566edd28e599b59a3be80405bedf150e5d5dd4d0f3d1d84e8e7f7dfe945d358493bb119470aabbf5b3eb720
5f2184793e5f9b41c0c622ebf0b83730feb3c2d06de8f7992cee469bf104595cd98307f7d3d584760b6f1d75e1
0f5132562662705 0f5b636f34b93471c290b1afede150055b796d3d08f31b52360ef28630acc273bfcda6aa14d
f76867cc230c0597bd76105d7a54426698d87ad1ac683c7569b80f0f4c5d169dd0dbcf390e8449b698649183cd
f214a13e51ed5c5c38df9931ad23a05e49b85975bcc65cca5ebfb404f62552cf46fd535e5d9e8178dff0156fbd
6227e2e04c56af73e8a149dbd63f5cd0a5ec1046c30b3a25ba60cfc869df084553430e7058e948b8e426003421
988789c266fcf9f6c309fcc785e11e76121316f82b61555352c91f3d936dbc1e181a6924d480ba09a62adf930d
ee5884ce5362ff31f1e2a4558702bc0d8c871cc322efad66efc946c3a9ae959ef20c052787d6a5e04d7dc9dfc2
c9941104ad26c136a9827a866b9e0942dbacd4aed56b48547c6dc1d0216fcdd2b5ce40828bdae5df48e724232b
01cc567173e07b9089e7834bb92c873c5e08ba055698df5f79ee73e122b5b72ee3e2e100858dad409da55ad0fa
1aa9caa60bf9c25e9ba3e1dc012724c89903a820b63dd5f14f0019007b180684afbe125a0cf87af796ae20e465
641c5e8fb91b6e7c6395d2f49f17bac3e35110c16b119bf289e11afb4bbae40266aa87605298ec5bb0601fa415
038621ea100db5d9d8a5da4d74fd92ed882546afc7a8a3dc648b1c7852e8fa43ca9ed287a4bb9dd299bfc69414
f990bf3b7b58a932a4ff9c86e2fb131b7cc7b65464cc80011267ab49f5c599b9ce43acc9a06b856b25b6fe5f51
afd6db147386a1ab30575a0eae607ffe116cae7fb70df6841ae0f52eaf359305becdf7f4636f3c31fab45387ef
97cef0b8ef6a8de702a2f1c21c3aee10382e8b8610a03bf6f1c54527a1d6db79f80f71c7f29880bb4b0d46cee3
a6063028e901f2f3514639bce237259a3122beea38763e205263f663543bf9bb390337f6a992b74dbfab966a9a
bf3dd54b15956b47ee731d23ec9a87a0027ed16f6fc4384812df153231c61331ef22e16035719044fddd7b6a96
26aa8397073d0cbf42fffd2b8201dca65aadaf98048e7bc0917ed83c7cbb59086ceff885b523fca2d65d87323f
d6bbd75b0a312b317b217985334086a63cdc4f065be41dfc590c76d71f93f76c091ee6b2878c2d718adde4493e
ef3a4cb8b2764e7013b80f922bf2b3b3572029975bc1388b7c975f353a02fa01ab679bbe041041dc1425787606
df44013b7627c4d43b1edbda3768386f2b9a78470ac7ac5ca6b2d20385d53e2e412828e2ebcdd17ce8658c655b
abeefe0b2f43f606a192613263a2c27831a7c5953c9a1f2910093b4720e78e3b34ae7102e1094fcc6655d3af6d
9a4c8f58f2478ef532121b7c91fe558a80257d63d8e9b4300c09dc04d9d37975f5cf3e548f23e40c070822419a
74267d08453815ba8e02943c3bc5a13ad669fd79d8e4ee5f5be82dc638ba9a1d
```

Session termination:

```
a16673746174757314
```

In diagnostic notation:

SessionEstablishment:

```
{
  "eReaderKey":
  24(<<
    {
      1: 2,
      -1: 1,
      -2: h'60E3392385041F51403051F2415531CB56DD3F999C71687013AAC6768BC8187E',
      -3: h'E58DEB8FDBE907F7DD5368245551A34796F7D2215C440C339BB0F7B67BECCDFA'
    }
  >>),
  "data": h'52AD ..redacted for readability.. E67C'
}
```

SessionData:

```
{
  "data": h'46DA ..redacted for readability.. 9A1D'
}
```

Session termination:

```
{
"status": 20
}
```

The following intermediate cryptographic data was used:

```
SKReader Zab: 6423502f843d8cda01fbd9fa46cb397534a740ab1ec3d1076fbcb12e1dca2589
SKReader key: 58d277d8719e62a1561d248f403f477e9e6c37bf5d5fc5126f8f4c727c22dfc9
SKDevice Zab: 6423502f843d8cda01fbd9fa46cb397534a740ab1ec3d1076fbcb12e1dca2589
SKDevice key: 81d170e07fbdac93c1a676242c2576124a380d87bb73ed9ce4834de2272cf409

SessionTranscriptBytes:
d81859024183d8185858a20063312e30018201d818584ba4010220012158205a88d182bce5f42efa59943f3335
9d2e8a968ff289d93e5fa444b624343167fe225820b16e8cf858ddc7690407ba61d4c338237a8cfcf3de6aa672
fc60a557aa32fc67d818584ba40102200121582060e3392385041f51403051f2415531cb56dd3f999c71687013
aac6768bc8187e225820e58deb8fdbe907f7dd5368245551a34796f7d2215c440c339bb0f7b67beccdfa8258c3
91020f487315d10209616301013001046d646f631a200c016170706c69636174696f6e2f766e642e626c756574
6f6f74682e6c652e6f6f6230081b28128b37282801021c015c1e580469736f2e6f72673a31383031333a646576
696365656e676167656d656e746d646f63a20063312e30018201d818584ba4010220012158205a88d182bce5f4
2efa59943f33359d2e8a968ff289d93e5fa444b624343167fe225820b16e8cf858ddc7690407ba61d4c338237a
8cfcf3de6aa672fc60a557aa32fc6758cd91022548721591020263720102110204616301013000110206616301
036e666300510204616301015700 1a201e016170706c69636174696f6e2f766e642e626c7565746f6f74682e6c
652e6f6f6230081b28078080bf2801021c021107c832fff6d26fa0beb34dfcd555d4823a1c11010369736f2e6f
72673a31383031333a6e66636e6663015a172b016170706c69636174696f6e2f766e642e7766612e6e616e5703
0101032302001324fec9a70b97ac9684a4e326176ef5b981c5e8533e5f00298cfccbc35e700a6b020414
```

D.5.2 Issuer data authentication

Taking the examples from D.3.2, D.3.3, D.4.1.1, D.4.1.2, D.5.1 and the following data:

IACA:

```
308201ce30820173a00302010202142ab4edd052b2582f4c6ad96186de70f4de5a3994300a06082a8648ce3d04
030230233114301206035504030c0b75746f706961206961636131 0b3009060355040613025553301e170d3230
3130303130303030305a170d3239303932393030303030305a30233114301206035504030c0b75746f706961
2069616361310b3009060355040613025553305930130607 2a8648ce3d020106082a8648ce3d03010703420004
2c3e103dbc07b25c5a770aeedfa5d8bd15417e3e676142461a7875e3b4188a2221e6423599d1db19aaef66f923
d394b61709549bcec2ea6ff60ec75268f2e094a38184308181301e0603551d120417301581136578616d706c65
406578616d706c652e636f6d301c0603551d1f041530133011a00fa00d820b6578616d706c652e636f6d301d06
03551d0e0416041454fa2383a04c28e0d930792261c80c4881d2c00b300e0603551d0f0101ff04040302010630
120603551d130101ff040830060101ff020100300a06082a8648ce3d0403020349003046022100ec897f0b8ae5
1028288955031f860069659b75989af7129fa609c24299a5c787022100d088d8741f5d05b360ef6e85023e90df
1d31dd1e6701a88efe9a7103021f986c
```

NOTE 1 The payload of a COSE_Sign1 structure is of type bstr. In the COSE_Sign1 structure, the `MobileSecurityObjectBytes`, which is a tagged CBOR bytestring, is therefore itself encapsulated in a bytestring.

NOTE 2 Note the possible options for the encoding types of the `x5chain` element as defined in RFC 9360.

IssuerAuth CBOR data:

```
8443a10126a118215901f3308201ef30820195a00302010202143c4416eed784f3b413e48f56f075abfa6d87eb
84300a06082a8648ce3d04030230233114301206035504030c0b75746f7069612069616361310b300906035504
0613025553301e170d3230313030313030303030305a170d3231313030313030303030305a302131123010060З
5504030c0975746f706961206473310b30090603550406130255533059301306072a8648ce3d020106082a8648
ce3d03010703420004ace7ab7340e5d9648c5a72a9a6f56745c7aad436a03a43efea77b5fa7b88f0197d57d898
3e1b37d3a539f4d588365e38cbbf5b94d68c547b5bc8731dcd2f146ba381a83081a5301e0603551d1204173015
81136578616d706c65406578616d706c652e636f6d301c0603551d1f041530133011a00fa00d820b6578616d70
6c652e636f6d301d0603551d0e0416041414e29017a6c35621ffc7a686b7b72db06cd12351301f0603551d2304
183016801454fa2383a04c28e0d930792261c80c4881d2c00b300e0603551d0f0101ff04040302078030150603
551d250101ff040b3009060728818c5d050102300a06082a8648ce3d0403020348003045022100977 17ab90167
40c8d7bcdaa494a62c053bbdecce1383c1aca72ad08dbc04cbb202203bad859c13a63c6d1ad67d814d43e2425c
af90d422422c04a8ee0304c0d3a68d5903a2d81859039da66776657273696f6e63312e306f646967657374416c
676f726974686d675348412d3235366c76616c756544696765737473a2716f72672e69736f2e31383031332e35
2e31ad00582075167333b47b6c2bfb86eccc1f438cf57af055371ac55e1e359e20f254adcebf01582067e539d6
139ebd131aef441b445645dd831b2b375b390ca5ef6279b205ed45710258203394372ddb78053f36d5d869780e
61eda313d44a392092ad8e0527a2fbfe55ae0358202e35ad3c4e514bb67b1a9db51ce74e4cb9b7146e41ac52da
c9ce86b8613db555045820ea5c3304bb7c4a8dcb51c4c13b65264f845541341342093cca786e058fac2d590558
20fae487f68b7a0e87a749774e56e9e1dc3a8ec7b77e490d21f0e1d3475661aa1d0658207d83e507ae77db815d
e4d803b88555d0511d894c897439f5774056416a1c7533075820f0549a145f1cf75cbeeffa881d4857dd438d62
7cf32174b1731c4c38e12ca936085820b68c8afcb2aaf7c581411d2877def155be2eb121a42bc9ba5b7312377e
068f660958200b3587d1dd0c2a07a35bfb120d99a0abfb5df56865bb7fa15cc8b56a66df6e0c0a5820c98a170c
f36e11abb724e98a75a5343dfa2b6ed3df2ecfbb8ef2ee55dd41c8810b5820b57dd036782f7b14c6a30faaaae6
ccd5054ce88bdfa51a016ba75eda1edea9480c5820651f8736b18480fe252a03224ea087b5d10ca5485146c67c
74ac4ec3112d4c3a746f72672e69736f2e31383031332e352e312e5553a4005820d80b83d25173c484c5640610
ff1a31c949c1d934bf4cf7f18d5223b15dd4f21c0158204d80e1e2e4fb246d97895427ce7000bb59bb24c8cd00
3ecf94bf35bbd2917e340258208b331f3b685bca372e85351a25c9484ab7afcdf0d2233105511f778d98c2f544
035820c343af1bd1690715439161aba73702c474abf992b20c9fb55c36a336ebe01a876d6465766963654b6579
496e666fa1696465766963654b6579a4010220012158209 6313d6c63e24e3372742bfdb1a33ba2c897dcd68ab8
c753e4fbd48dca6b7f9a2258201fb3269edd418857de1b39a4e4a44b92fa484caa722c228288f01d0c03a2c3d6
67646f6354797065756f72672e69736f2e31383031332e352e312e6d444c6c76616c6964697479496e666fa366
7369676e6564c074323032302d31302d30315431333a33303a30325a6976616c696446726f6dc074323032302d
31302d30315431333a33303a30325a6a76616c6964556e74696cc074323032312d31302d30315431333a33303a
30325a584059e64205df1e2f708dd6db0847aed79fc7c0201d80fa55badcaf2e1bcf5902e1e5a62e4832044b89
0ad85aa53f129134775d733754d7cb7a413766aeff13cb2e
```

In diagnostic notation:

```
[
  << {1: -7} >>,
  {
    33:
h'308201EF30820195A00302010202143C4416EED784F3B413E48F56F075ABFA6D87EB84300A06082A8648CE3D
0403023023311430120603550 4030C0B75746F706961206961636131 0B3009060355040613025553301E170D32
30313030313030303030305A170D3231313030313030303030305A302131123010060355040 30C0975746F7069
61206473310B30090603550406130255533059301306072A8648CE3D020106082A8648CE3D03010703420004AC
E7AB7340E5D9648C5A72A9A6F56745C7AAD436A03A43EFEA77B5FA7B88F0197D57D8983E1B37D3A539F4D58836
5E38CBBF5B94D68C547B5BC8731DCD2F146BA381A83081A5301E0603551D120417301581136578616D706C6540
6578616D706C652E636F6D301C0603551D1F041530133011A00FA00D820B6578616D706C652E636F6D301D0603
551D0E0416041414E29017A6C35621FFC7A686B7B72DB06CD12351301F0603551D2304183016801454FA2383A0
4C28E0D930792261C80C4881D2C00B300E0603551D0F0101FF04040302078030150603551D250101FF040B3009
060728818C5D050102300A06082A8648CE3D0403020348003045022100977 17AB9016740C8D7BCDAA494A62C05
3BBDECCE1383C1ACA72AD08DBC04CBB202203BAD859C13A63C6D1AD67D814D43E2425CAF90D422422C04A8EE03
04C0D3A68D'
  },
  <<
```

```
    24(<<
      {
        "version": "1.0",
        "digestAlgorithm": "SHA-256",
        "valueDigests":
        {
          "org.iso.18013.5.1":
          {
            0: h'75167333B47B6C2BFB86ECCC1F438CF57AF055371AC55E1E359E20F254ADCEBF',
            1: h'67E539D6139EBD131AEF441B445645DD831B2B375B390CA5EF6279B205ED4571',
            2: h'3394372DDB78053F36D5D869780E61EDA313D44A392092AD8E0527A2FBFE55AE',
            3: h'2E35AD3C4E514BB67B1A9DB51CE74E4CB9B7146E41AC52DAC9CE86B8613DB555',
            4: h'EA5C3304BB7C4A8DCB51C4C13B65264F845541341342093CCA786E058FAC2D59',
            5: h'FAE487F68B7A0E87A749774E56E9E1DC3A8EC7B77E490D21F0E1D3475661AA1D',
            6: h'7D83E507AE77DB815DE4D803B88555D0511D894C897439F5774056416A1C7533',
            7: h'F0549A145F1CF75CBEEFFA881D4857DD438D627CF32174B1731C4C38E12CA936',
            8: h'B68C8AFCB2AAF7C581411D2877DEF155BE2EB121A42BC9BA5B7312377E068F66',
            9: h'0B3587D1DD0C2A07A35BFB120D99A0ABFB5DF56865BB7FA15CC8B56A66DF6E0C',
            10: h'C98A170CF36E11ABB724E98A75A5343DFA2B6ED3DF2ECFBB8EF2EE55DD41C881',
            11: h'B57DD036782F7B14C6A30FAAAAE6CCD5054CE88BDFA51A016BA75EDA1EDEA948',
            12: h'651F8736B18480FE252A03224EA087B5D10CA5485146C67C74AC4EC3112D4C3A'
          },
          "org.iso.18013.5.1.US":
          {
            0: h'D80B83D25173C484C5640610FF1A31C949C1D934BF4CF7F18D5223B15DD4F21C',
            1: h'4D80E1E2E4FB246D97895427CE7000BB59BB24C8CD003ECF94BF35BBD2917E34',
            2: h'8B331F3B685BCA372E85351A25C9484AB7AFCDF0D2233105511F778D98C2F544',
            3: h'C343AF1BD1690715439161ABA73702C474ABF992B20C9FB55C36A336EBE01A87'
          }
        },
        "deviceKeyInfo":
        {
          "deviceKey":
          {
            1: 2,
            -1: 1,
            -2: h'96313D6C63E24E3372742BFDB1A33BA2C897DCD68AB8C753E4FBD48DCA6B7F9A',
            -3: h'1FB3269EDD418857DE1B39A4E4A44B92FA484CAA722C228288F01D0C03A2C3D6'
          }
        },
        "docType": "org.iso.18013.5.1.mDL",
        "validityInfo":
        {
          "signed": 0("2020-10-01T13:30:02Z"),
          "validFrom": 0("2020-10-01T13:30:02Z"),
          "validUntil": 0("2021-10-01T13:30:02Z")
        }
      }
    >>)
  >>,
  h'59E64205DF1E2F708DD6DB0847AED79FC7C0201D80FA55BADCAF2E1BCF5902E1E5A62E4
832044B890AD85AA53F129134775D733754D7CB7A413766AEFF13CB2E'
]
```

The following intermediate cryptographic data was used:

```
Sig structure:
846a5369676e61747572653143a10126405903a2d81859039da66776657273696f6e63312e306f646967657374
416c676f726974686d675348412d3235366c76616c756544696765737473a2716f72672e69736f2e3138303133
```

```
2e352e31ad00582075167333b47b6c2bfb86eccc1f438cf57af055371ac55e1e359e20f254adcebf01582067e5
39d6139ebd131aef441b445645dd831b2b375b390ca5ef6279b205ed45710258203394372ddb78053f36d5d869
780e61eda313d44a392092ad8e0527a2fbfe55ae0358202e35ad3c4e514bb67b1a9db51ce74e4cb9b7146e41ac
52dac9ce86b8613db555045820ea5c3304bb7c4a8dcb51c4c13b65264f845541341342093cca786e058fac2d59
055820fae487f68b7a0e87a749774e56e9e1dc3a8ec7b77e490d21f0e1d3475661aa1d0658207d83e507ae77db
815de4d803b88555d0511d894c897439f5774056416a1c7533075820f0549a145f1cf75cbeeffa881d4857dd43
8d627cf32174b1731c4c38e12ca936085820b68c8afcb2aaf7c581411d2877def155be2eb121a42bc9ba5b7312
377e068f660958200b3587d1dd0c2a07a35bfb120d99a0abfb5df56865bb7fa15cc8b56a66df6e0c0a5820c98a
170cf36e11abb724e98a75a5343dfa2b6ed3df2ecfbb8ef2ee55dd41c8810b5820b57dd036782f7b14c6a30faa
aae6ccd5054ce88bdfa51a016ba75eda1edea9480c5820651f8736b18480fe252a03224ea087b5d10ca5485146
c67c74ac4ec3112d4c3a746f72672e69736f2e31383031332e352e312e5553a4005820d80b83d25173c484c564
0610ff1a31c949c1d934bf4cf7f18d5223b15dd4f21c0158204d80e1e2e4fb246d97895427ce7000bb59bb24c8
cd003ecf94bf35bbd2917e340258208b331f3b685bca372e85351a25c9484ab7afcdf0d2233105511f778d98c2
f544035820c343af1bd1690715439161aba73702c474abf992b20c9fb55c36a336ebe01a876d6465766963654b
6579496e666fa1696465766963654b6579a4010220012158209 6313d6c63e24e3372742bfdb1a33ba2c897dcd6
8ab8c753e4fbd48dca6b7f9a2258201fb3269edd418857de1b39a4e4a44b92fa484caa722c228288f01d0c03a2
c3d667646f6354797065756f72672e69736f2e31383031332e352e312e6d444c6c76616c6964697479496e666f
a3667369676e6564c074323032302d31302d30315431333a33303a30325a6976616c696446726f6dc074323032
302d31302d30315431333a33303a30325a6a76616c6964556e74696cc074323032312d31302d30315431333a33
303a30325a
```

D.5.3 mdoc authentication

Taking the examples from D.3.2, D.3.3, D.4.1.1, D.4.1.2, D.5.1, D.5.2 and the following data:

Static device key:

```
  x: 96313d6c63e24e3372742bfdb1a33ba2c897dcd68ab8c753e4fbd48dca6b7f9a
  y: 1fb3269edd418857de1b39a4e4a44b92fa484caa722c228288f01d0c03a2c3d6
  d: 6ed542ad4783f0b18c833fadf2171273a35d969c581691ef704359cc7cf1e8c0
```

DeviceAuth CBOR data:

```
a16964657669636541d61638443a10105a0f65820e99521a85ad7891b806a07f8b5388a332d92c189a7bf293ee1
f543405ae6824d
```

In diagnostic notation:

```
{
  "deviceMac":
  [
    << {1: 5} >>,
    {},
    null,
    h'E99521A85AD7891B806A07F8B5388A332D92C189A7BF293EE1F543405AE6824D'
  ]
}
```

The following intermediate cryptographic data was used:

```
EMacKey Zab: 78d98a86fbbb82895874bfafcc161ba69f9b77662172c74b3b0d4643276cf991
EMacKey key: dc2b9566fdaaae3c06baa40993cd0451aeba15e7677ef5305f6531f3533c35dd
MAC tag: e99521a85ad7891b806a07f8b5388a332d92c189a7bf293ee1f543405ae6824d
DeviceAuthenticationBytes:
d818590271847444657669636541757468656e7469636174696f6e83d8185858a20063312e30018201d818584b
a4010220012158205a88d182bce5f42efa59943f33359d2e8a968ff289d93e5fa444b624343167fe225820b16e
8cf858ddc7690407ba61d4c338237a8cfcf3de6aa672fc60a557aa32fc67d818584ba40102200121582060e339
2385041f51403051f2415531cb56dd3f999c71687013aac6768bc8187e225820e58deb8fdbe907f7dd53682455
51a34796f7d2215c440c339bb0f7b67beccdfa8258c391020f487315d10209616301013001046d646f631a200c
016170706c69636174696f6e2f766e642e626c7565746f6f74682e6c652e6f6f6230081b28128b37282801021c
015c1e580469736f2e6f72673a31383031333a646576696365656e676167656d656e746d646f63a20063312e30
018201d818584ba4010220012158205a88d182bce5f42efa59943f33359d2e8a968ff289d93e5fa444b6243431
67fe225820b16e8cf858ddc7690407ba61d4c338237a8cfcf3de6aa672fc6758cd910225487215
91020263720102110204616301013000110206616301036e666300510204616301015700 1a201e016170706c69
```

```
636174696f6e2f766e642e626c7565746f6f74682e6c652e6f6f6230081b28078080bf2801021c021107c832ff
f6d26fa0beb34dfcd555d4823a1c11010369736f2e6f72673a31383031333a6e66636e6663015a172b01617070
6c69636174696f6e2f766e642e7766612e6e616e57030101032302001324fec9a70b97ac9684a4e326176ef5b9
81c5e8533e5f00298cfccbc35e700a6b020414756f72672e69736f2e31383031332e352e312e6d444cd81841a0

MAC structure:
84644d41433043a1010540590276d8185902718474446576696365417574686656e7469636174696f6e83d81858
58a20063312e30018201d818584ba4010220012158205a88d182bce5f42efa59943f33359d2e8a968ff289d93e
5fa444b624343167fe225820b16e8cf858ddc7690407ba61d4c338237a8cfcf3de6aa672fc60a557aa32fc67d8
18584ba40102200121582060e3392385041f51403051f2415531cb56dd3f999c71687013aac6768bc8187e2258
20e58deb8fdbe907f7dd5368245551a34796f7d2215c440c339bb0f7b67beccdfa8258c391020f487315d10209
616301013001046d646f631a200c016170706c69636174696f6e2f766e642e626c7565746f6f74682e6c652e6f
6f6230081b28128b37282801021c015c1e580469736f2e6f72673a31383031333a646576696365656e67616765
6d656e746d646f63a20063312e30018201d818584ba4010220012158205a88d182bce5f42efa59943f33359d2e
8a968ff289d93e5fa444b624343167fe225820b16e8cf858ddc7690407ba61d4c338237a8cfcf3de6aa672fc60
a557aa32fc6758cd91022548721591020263720102110204616301013000110206616301036e66630051020461
63010157001a201e016170706c69636174696f6e2f766e642e626c7565746f6f74682e6c652e6f6f6230081b28
078080bf2801021c021107c832fff6d26fa0beb34dfcd555d4823a1c11010369736f2e6f72673a31383031333a
6e66636e6663015a172b016170706c69636174696f6e2f766e642e7766612e6e616e57030101032302001324fe
c9a70b97ac9684a4e326176ef5b981c5e8533e5f00298cfccbc35e700a6b020414756f72672e69736f2e313830
31332e352e312e6d444cd81841a0
```

D.5.4 mdoc reader authentication

Taking the examples from D.3.2, D.3.3, D.4.1.1, D.4.1.2, D.5.1 and the following data:

Reader root:

```
3082019030820137a003020102021430d747795405d564b7ac48be6f364ae2c774f2fc300a06082a8648ce3d04
0302301631143012060355040300c0b72656164657220726f6f74301e170d32303130303130303030305a170d
32393039323930303030305a301631143012060355040300c0b72656164657220726f6f7430593013060072a86
48ce3d020106082a8648ce3d03010703420004364329383 2e0a480de592df0708fe25b6b923f6397ab39a8b1b7
444593adb89c77b7e9c28cf48d6d187b43c9bf7b9c2c5c5ef22f329e44e7a91b4745b7e2063aa3633061301c06
03551d1f041530133011a00fa00d820b6578616d706c652e636f6d301d0603551d0e04160414cfb7a881baea5f
32b6fb91cc29590c50dfac416e300e0603551d0f0101ff04040302010630120603551d130101ff040830060101
ff020100300a06082a8648ce3d0403020347003044022018ac84baf991a614fb25e76241857b7fd0579dfe8aed
8ac7f130675490799930022077f46f00b4af3e014d253e0edcc9f146a75a6b1bdfe33e9fa72f30f0880d5237
```
ReaderAuth CBOR data:

```
8443a10126a118215901b7308201b330820158a00302010202147552715f6add323d4934a1ba175dc945755d8b
50300a06082a8648ce3d04030230163114301206035504030c0b72656164657220726f6f74301e170d32303130
30313030303030305a170d32333132333130303030305a3011310f300d06035504030c06726561646572305 9
301306072a8648ce3d020106082a8648ce3d03010703420004f8912ee0f912b6be683ba2fa0121b2630e601b2b
628dff3b44f6394eaa9abdbcc2149d29d6ff1a3e091135177e5c3d9c57f3bf839761eed02c64dd82ae1d3bbfa3
8188308185301c0603551d1f041530133011a00fa00d820b6578616d706c652e636f6d301d0603551d0e041604
14f2dfc4acafc5f30b464fada20bfcd533af5e07f5301f0603551d23041830168014cfb7a881baea5f32b6fb91
cc29590c50dfac416e300e0603551d0f0101ff04040302078030150603551d250101ff040b3009060728818c5d
050106300a06082a8648ce3d0403020349003046022100fb9ea3b686fd7ea2f0234858ff8328b4efef6a1ef71e
c4aae4e307206f921493022100 9b94f0d739dfa84cca29efed529dd4838acfd8b6bee212dc6320c46feb839a35
f658401f3400069063c189138bdcd2f631427c589424113fc9ec26cebcacacfcdb9695d28e99953becabc4e30a
b4efacc839a81f9159933d192527ee91b449bb7f80bf
```
In diagnostic notation:

```
[
  << {1: -7} >>
  {
    33:
308201B330820158A00302010202147552715F6ADD323D4934A1BA175DC945755D8B50300A06082A8648CE3D04
0302301631143012060355040300C0B72656164657220726F6F74301E170D32303130303130303030305A170D
```

```
323331323331303030303030305A3011310F300D06035504030C067265616465723059301306072A8648CE3D0201
```

```
MSO:
{
    ...
    "status": {
        "identifier_list": {
            "id": h'cccc',
            "uri": "https://example.com/identifierlists/1",
            "certificate": h'aa...'
        }
    }
}

Identifier list:
18([
    << {1: -7, 16: "identfierlist+cwt", 33:h'aa....'} >>,
    {},
    << {
        2: "https://example.com/identifierlists/1",
        4: 2291720170,
        6: 1686920170,
        65534: 86400
        65530: {
                "identifiers": {
                    h'abcd': {},
                    h'aaaa': {},
                    h'cccc': {},
                }
                "aggregation_uri":
"https://example.com/identifierlists/aggregation"},
    } >>,
    h'aa....'
])
```

Status list example:

```
MSO:
{
    ...
    "status": {
        "status_list": {
            "idx": 1340,
            "uri": "https://example.com/statuslists/1",
            "certificate": h'aa...'
        }
    }
}

Status list:
18([
    << {1: -7, 16: "statuslist+cwt", 33:h'aa....'} >>,
    {},
    << {
        2: "https://example.com/statuslists/1",
        4: 2291720170,
        6: 1686920170,
        65534: 86400
        65533: {
            "bits": 1
            "lst": h'aa....'
            "aggregation_uri": "https://example.com/statuslists/aggregation"},
    } >>,
    h'aa....'
])
```

D.7 Examples of data signing using DeviceSigned

D.7.1 Overview

Data elements can either be signed by the issuer at the time of using the mdoc or by the device at transaction time. One use case for using the data elements signed by the device is to sign data provided by the mdoc verifier.

To provide data to the mdoc, the `DocRequestInfo` extension point in the request can be used. This subclause contains examples that show the concept of using `DocRequestInfo` and `DeviceSigned` to sign data provided by the mdoc verifier. These examples are intended only to show the concepts and are not designed to be used as-is.

The contents of the `DocRequestInfo` element, what exactly is returned in the `DeviceSigned` data element, what data elements to use to provide back the signed data are all use case specific and out of scope for this document.

D.7.2 Signing of a document

One example of signing data is the signing of a document. In this example the document or link to a document is provided in the request. The mdoc then returns a signature over the hash of the document in the response.

In this example, the profile defines the following:

The identifier for the `RequestInfo` element is “com.example.docsigninfo”, the structure is a map that contains either the document itself as bytestring or a URI pointing to the document. The identifier of the data element used to return the signed data is “docsign” and the value of that data element is defined as containing the sha-256 hash as a bstr.

Example of the request in diagnostic notation:

```
{
  "version": "1.0",
  "docRequests": [
    {
      "itemsRequest":
      24(<<
        {
          "docType": "com.example.doctype",
          "nameSpaces": {
            "com.example.namespace": {
              "first_name": true,
              "last_name": true,
              "document_number": true,
              "issue_date": true,
              "expiry_date": true
              "docsign": true
            }
          }
          "requestInfo": {
            "com.example.docsigninfo": {
                "document": h'6578616d706c65..'
            }
          }
        }
      )
    }
  ]
}
```

The response then contains “first_name”, “last_name”, “document_number”, “issue_date” and “expiry_date” as `IssuerSigned` data elements and the “docsign” data element as `DeviceSigned` data element. Note that as described in 12.3.4, it is always required for the DeviceSigned data element to be authorized in the MSO using the `KeyAuthorizations` structure. An example of a response just focussing on the parts relevant to the DeviceSigned element looks like the following in diagnostic notation:

MSO:

```
  {
      ...
      "deviceKeyInfo": {
          "deviceKey": {
              1: 2,
              -1: 1,
              -2: h'70911be02958150bc648cc7acbddf3cfed0187599d14ed34703216982e0403a0',
              -3: h'9df502d8273af24ea39cc471ab1d52ee1abe8d18f2a2bd01f6e6f766ef1702c3'
          },
          "keyAuthorizations": {
              "dataElements": {
                  "com.example.namespace": [
                      "docsign"
                  ]
              }
          }
      }
  }
```

DeviceSigned:

```
  "deviceSigned": {
      "nameSpaces": 24(<< {
          "com.example.namespace": {
              "docsign":
h'd373fc001cb1bf1a1eec89ecaaec6ba0599bb75efa67b2a44db907e7fcf7fb02'
          }
      } >>),
      "deviceAuth": {
        "deviceMac":
        [
          << {1: 5} >>,
          {},
          null,
          h'6d027363b47e215d7bcd1f44c44431071dbc8ce39c07009954815f4aa0bfa266'
        ]
      }
  }
```

D.7.3 Signing of a payment information

In this example payment information is provided in the request. The mdoc then builds up a payfment confirmation structure and includes it in the response.

In this example, the profile defines the following:

The identifier for the `RequestInfo` element is "com.example.paymentinfo", the structure is a map that contains rudimentary payment info to be confirmed. The identifier of the data element used to return the signed data is "payment" and the value of that data element contains a very rudimentary example of payment confirmation by the mdoc.

Example of the request in diagnostic notation:

```
{
  "version": "1.0",
  "docRequests": [
    {
      "itemsRequest":
      24(<<
        {
          "docType": "com.example.paymentcredential",
          "nameSpaces": {
            "com.example.namespace": {
              "account_number": true,
              "issue_date": true,
              "expiry_date": true
              "payment": true
            }
          }
          "requestInfo": {
            "com.example.paymentinfo": {
                "amount": 20.2,
                "currency": "usd"
                "merchant": "example merchant"
            }
          }
        }
      )
    }
  ]
}
```

The response then contains "first_name", "last_name", "document_number", "issue_date" and "expiry_date" as `IssuerSigned` data elements and the "docsign" data element as `DeviceSigned` data element. Note that as described in 12.3.4, it is always required for the DeviceSigned data element to be authorized in the MSO using the `KeyAuthorizations` structure. An example of a response just focussing on the parts relevant to the DeviceSigned element looks like the following in diagnostic notation:

MSO:

```
{
    ...
    "deviceKeyInfo": {
        "deviceKey": {
            1: 2,
            -1: 1,
            -2: h'70911be02958150bc648cc7acbddf3cfed0187599d14ed34703216982e0403a0',
            -3: h'9df502d8273af24ea39cc471ab1d52ee1abe8d18f2a2bd01f6e6f766ef1702c3'
        },
        "keyAuthorizations": {
            "dataElements": {
                "com.example.namespace": [
                    "payment"
                ]
            }
        }
    }
}
```

DeviceSigned:

```
"deviceSigned": {
    "nameSpaces": 24(<< {
        "com.example.namespace": {
            "payment": {
                "amount": 20.2,
                "currency": "usd",
                "merchant": "example merchant",
            }
        }
    } >>),
    "deviceAuth": {
      "deviceMac":
      [
        << {1: 5} >>,
        {},
        null,
        h'e0379960afe4eab7a72ac00a32c1fd2434a998ba4732810df0d7925e6b8e956b'
      ]
    }
}
```

Annex E
(informative)

Privacy and security recommendations

E.1 General

This annex provides guidance for issuing authorities to design and implement their mDL solutions for privacy and security. It also contains background information on privacy, and a section on how the existing standard for data transfer has already addressed privacy concerns.

E.2 Achieving privacy for the mDL holder

E.2.1 Privacy goals

Individual privacy and security of personally identifiable information (PII) in the mobile, electronic age must be ensured and is a shared responsibility of all involved parties. No technical standard for data interchange can dictate how all privacy measures are achieved. Privacy is achieved by the end-to-end solution, and with the participation of all participants in an ecosystem. Each actor in the mDL ecosystem should fulfill their role in a responsible manner that best protects PII.

E.2.2 Privacy definitions, goals, and principles

This document has been developed with the following privacy-by-design goals and allows both mDL holder and issuing authority concerns listed in this annex to be achieved in multiple ways:

— data minimization and anonymization wherever possible;

— be proactive to prevent data breach;

— privacy should be the default setting;

— embed privacy in design, flows, and architecture;

— privacy does not need to be traded off for full functionality;

— protect the full lifecycle of the identity end-to-end;

— keep all operations visible and transparent to the mDL holder;

— design for user-centricity and user-control of their identity.

To achieve these privacy-by-design goals, ISO/IEC 29100 defines the following principles for privacy protection:

1. consent and choice,
2. purpose legitimacy and specification,
3. collection limitation,
4. data minimization,

5. use, retention and disclosure limitation,
6. accuracy and quality,
7. openness, transparency and notice,
8. individual participation and access,
9. accountability,
10. information security,
11. privacy compliance.

These principles put names on the protections that consumers (data subject) expect from data controllers and their data processors. These have been codified into law (e.g. GDPR) or regional principles (e.g. US DHS Fair Information Practice Principles).

— **Consent and choice** – The data subject must consent to the processing of their personal data (see definitions in the section on mDL holder Consent below).

— **Purpose legitimacy and specification** – The data subject should be fully aware of the purpose for which their personal data is being collected, processed, and potentially stored.

— **Collection limitation** – The data controller and data processors should only collect the data necessary for their purpose and should only collect data consistent with these principles.

— **Data minimization** – Processing of data should be minimized to that specifically necessary for the purpose specified.

— **Use, retention, and disclosure limitation** – Data processors should not use personal data of the data subject except for the purposes specified and consistent with these other principles. Personal data should only be retained for the period necessary to provide the service.

— **Accuracy and quality** – High accuracy of data being processed and held is in the best interest of the data subject and processors should take measures to ensure accuracy.

— **Openness, transparency and notice** – What data and how data is being processed should be well-known to the data subject, including obtaining consent, and posting and updating clear notices.

— **Individual participation and access** – The data subject should be involved in the collection, consent, processing, and storage management of their personal data.

— **Information security** – Personal data should be protected by security safeguards against such risks as loss or unauthorized access, destruction, use, modification or disclosure.

— **Privacy compliance** – The data controller and data processors must be accountable for all aspects of the processing of personal data and provide audit logs and auditability to the data subject.

See ISO/IEC 29100 for more information on these terms.

E.2.3 Privacy protections embedded in the protocol design

The privacy principles above were taken into consideration during protocol design, and a set of protections implemented at the protocol level in order to establish a privacy baseline. Table E.1 details protections embedded in the protocol. It lists protections that should be considered and implemented as part of solutions.

Table E.1 — Privacy principles and protections

Principle	Protection
4. Data minimization	Fields (data groups) have been separated into individual data elements to support privacy-preserving attribute queries. mDL readers request data elements individually and should ask for only those necessary for their use case.
3. Collection limitation 4. Data minimization	The request-response model permits disclosing additional data if the initial response did not fulfill the requirements of the use case. This lessens the need to over-ask for all data in the first pass.
1. Consent and choice	The data transfer model of this document provides the ability for mDL solutions to implement either pre-consent or transaction-time consent as defined below. Consent applies to both selective disclosure and authorization for intent-to-retain.
4. Data minimization	Optional mDL reader authentication may be used to restrict access to specific data elements of high sensitivity. This can prevent leakage, for example, of biometric data.
3. Collection limitation	Replay attacks and follow-up requests are not permitted once the connection is broken in either online or offline models. Use of “intent to retain” provides a framework for limitation of collection of data by an mDL reader.
10. Information security	mDL data is signed by the issuing authority using digital certificates for protection of integrity and authenticity. mDL readers use trusted certificates to validate the integrity and authenticity of mDL data.
8. Individual participation and access	Sequence of “Device engagement, secure connection, Request, Response, Repeat” permits pre-consent and transaction-time consent models at mDL holder discretion.
10. Information security (of data subject)	Recommendations are provided for setting up lower level transport protocols to minimize or prevent leakage of unique mDL holder or device identifiers. For Wi-Fi Aware and BLE, this document encourages random MAC addresses for mDL on every data path and encourages rotation of MAC address for service discovery. For NFC, no unique persistent identifiers should be made available at the protocol level. Higher layer addressing such as IPv6 addresses used in Wi-Fi Aware should also not leak unique mDL holder or device identifiers.
10. Information security (of data and data subject)	Ephemeral keys are used in channel security. mDL Data Authentication can prevent data tampering. Random numbers are used in Issuer Data Authentication signatures.
4. Data minimization	Privacy-preserving signatures: Cryptographic signatures do not leak additional unique identifiers. That is, the protocol does not make mDL holders identifiable if they cannot be identified by the transmitted data attributes themselves.
10. Information security (of data subject, unlinkability)	Multiple mDL transaction signatures and the MSO of a single mDL holder are distinct and not linkable. Linkability through mdoc authentication keys can be reduced by frequent updates of mdoc authentication keys.

6. Accuracy and quality	The data returned by this protocol in device retrieval is the one that was determined valid at the time of issuance of the MSO object. The validity element of the MSO allows the authority to limit the period during which this data will be considered sufficiently recent for offline use, and indicates whether an update of the MSO information was performed or not by the mDL application or mDL holder. The issuing authority should define appropriate periods of validity that balance freshness with offline capability, taking into account that a shorter validity period mitigates certain security risks.

E.2.4 Responsibilities of roles and participants

E.2.4.1 Participants in the mDL Ecosystem

Each responsible participant in the mDL Ecosystem should make technology choices in line with privacy principles and the recommendations in this annex. All participants should post their privacy policy in easily discoverable public locations and provide mechanisms for contact and redress of privacy problems encountered by individuals. The participants are the following:

— issuing authorities - entities that officially issue driving licences and now sign mDLs;

— technology providers - provide systems and Apps for issuing authorities to issue mDLs;

— mDL verifiers - operate mDL readers that obtain trustworthy data from mDLs;

— VICAL providers - publish lists of trustworthy IACA certificates for verifying data.

E.2.4.2 Issuing authorities

Above all, issuing authorities are responsible for selecting or developing technology that meets the privacy regulations of their jurisdictions and regions, and they have a responsibility to protect PII with the rigor with which they currently protect physical card issuances.

Issuing authorities are responsible for the accurate provisioning of mDL to the proper mDL holder to achieve their target Identity Assurance Level as defined within their regional trust frameworks (e.g. NIST SP 800-63). Issuing authorities are also responsible for security controls of the mDL in order to resist cloning and tampering including those binding the mDL to the mobile device, and to provide for the synchronization of relevant data updates to the mDL.

E.2.4.3 Technology providers

A technology provider should apply Privacy By Design principles for the applicable role under which they will provide software - issuing authority, mDL verifier, or Individual/Apps.

Technology providers should work with issuing authorities to ensure that mDLs provide adequate information about consent, including purpose, and clear indication of what data is needed for how long.

Similarly, providers of mDL verifier technology should work with mDL verifiers to ensure only the data required for each use case is requested during each transaction.

E.2.4.4 mDL verifiers (also known as relying parties)

Since the interchange of data with a Mobile Driving Licence is straightforward and easy to perform, mDL verifiers should never store PII, but rather request data from the mDL holder whenever it needs to process data to perform a transaction.

mDL verifiers should request the minimum data required for their use case. With each request for data, mDL verifiers should provide adequate informed consent, including the purpose and scope of data requested, to the mDL holder and never use blanket terms and conditions in lieu of informed consent.

E.2.4.5 VICAL providers

VICALs are signed lists of certificates from actual, real-world verified Issuing Authorities. These lists can be maintained by issuing authorities themselves, associations of issuing authorities, by vendors of mDL readers, or by others. The VICAL provider endeavors to keep the list of certificates current and extensive for their field of operation.

mDL verifiers should use mechanisms to assemble IACA certificates from known trusted Issuing Authorities. They can do it using solutions such as bilateral and/or regional agreements, in effect acting as their own VICAL provider or they may obtain VICALs of certificates from reliable VICAL providers.

E.2.4.6 mDL holders

Individuals should pay attention to each consent they may be asked to grant, and not to accept blanket terms and conditions or consent to share. Consent should not be granted by the mDL holder unless they are satisfied with the purpose for sharing specific data and know the mDL verifier with whom they are sharing.

E.3 Privacy recommendations

This annex contains guidance to assist issuing authorities and all participants in the mDL ecosystem so that they can release solutions that protect the privacy of their citizens and achieve the above privacy principles.

E.4 Transparency of data storage and use

mDL holders should have transparency into all data that is held in the mDL. The mDL holder should always be given the ability to consent to the sharing of that data and be informed of the onward storage of that data using the “intent to retain” flag. Proper informed consent as per Clause E.14 can help ensure transparency over the use of personal data to the mDL holder.

E.5 Protection of keys and mDL data

Device retrieval requires that mDLs be able to dynamically authenticate (sign or MAC) the data they return to an mDL reader in possession of the mDL verifier. This mDL authentication key should be protected from unauthorized usage.

The key should be protected with industry state-of-the-art protection methods. Examples are:

— a trusted execution environment;

— a hardware secure element with appropriate certifications;

— other secure hardware solution or combination of the aforementioned;

— use of biometrics or strong on-device authentication technology.

Controls and technology for fraud detection should be implemented.

In some other cases a backend could verify additional information such as digital signature, revocation and tokenization status. Such approach overall results in protecting the mDL holder by ensuring that the right mDL holder is presenting the data while not necessarily relying on the mDL holder device and mDL App security.

Many mDL data elements are PII and need also to be stored securely. If a secure area is available and encryption of such data elements is used, the encryption keys should be stored onto the secure area, regardless of data elements storage. Alternatively, data elements may be stored onto secure area with appropriate protection according to issuing authority security policy.

Implementations may further protect the mDL authentication key and data with a biometric or PIN.

E.6 DeviceSignedItems and IssuerSignedItems

For device retrieval, this document allows to return any data element as part of either IssuerSignedItems or DeviceSignedItems, see 10.3.3. This clause provides information on IssuerSignedItems and DeviceSignedItems with respect to security and privacy and provides recommendations for support.

If a data element is returned in IssuerSignedItems, it is digitally signed by the issuing authority as part of the mobile security object (MSO). The data element is returned by the mDL together with the MSO, which allows the mDL reader to verify the integrity and authenticity of the data element by verifying the digital signature of the MSO and comparing the hash value of each data element with the corresponding hash in the MSO.

In case of DeviceSignedItems, the data element is not digitally signed by the issuing authority itself. Instead, the issuing authority authorizes the mDL to return the data element together with a MAC or signature calculated over the data element using the mDL's mdoc authentication private key. The mDL returns the data element together with the mDL's MAC or signature and the MSO which includes the mDL's mdoc authentication public key as well as the issuing authority's authorization to return this data element as part of deviceSignedItems. The mDL reader can verify the integrity and authenticity of the data element by verifying the issuing authority’s digital signature of the MSO (including the mdoc authentication public key and the authorizations) and verifying the mDL's MAC or signature using the mdoc authentication public key.

Privacy considerations

From a privacy perspective the usage of IssuerSignedItems has the following issue compared to the usage of DeviceSignedItems: The data elements returned by an mDL as part of IssuerSignedItems together with the MSO can be validated by anybody thanks to the digital signature of the issuing authority over the data elements. If this data becomes public by accident or on purpose, any person or entity who comes into possession of this data has demonstrably authentic data. The mDL holder cannot dispute that this data is authentic. The use of issuerSignedItems allows any party in possession of this data, e.g. provided by a verifier or leaked by a compromised verifier, to validate the authenticity of this data or to demonstrate the authenticity of the data to any other party. Data elements returned by different mdocs (such as mDL, health certificates, training certificates etc.) of a person as part of IssuerSignedItems can be aggregated and result in a publicly verifiable profile of the mDL holder's data. If a verifier shares or leaks IssuerSignedItems together with the MSO, this data does not allow to track the verifier where the disclosure originated.

In contrast if the data elements are returned by an mDL as part of DeviceSignedItems and the mDL uses the mdoc MAC Authentication, only the mdoc reader involved in the transaction can verify the integrity and authenticity of the data, but no other entities. This does not apply if DeviceSignedItems are used with mdoc ECDSA / EdDSA Authentication.

Security considerations

From a security perspective the usage of DeviceSignedItems has the following issue compared to the usage of IssuerSignedItems: The integrity and authenticity of DeviceSignedItems relies not only on the security of the issuing authority's Document Signer and IACA private keys, as in the case of IssuerSignedItems, but also on the security of the mdoc authentication private key in the secure area of the mDL. If this key or control over the key is compromised, an attacker can not only clone the mDL (which is true for IssuerSignedItems as well as DeviceSignedItems), but can also assign any value to the data elements which the issuing authority authorized for use in DeviceSignedItems. If the authorization is given for the complete mDL namespace, an attacker could basically generate an arbitrary number of mDLs using arbitrary mDL data element values.

In addition, the integrity and authenticity of DeviceSignedItems relies on the secure handling of the corresponding mDL data elements in the mdoc. If an attacker can change the value of an mDL data element stored in the mdoc, a verifier will validate the DeviceSignedItems with the falsified mDL data. In case the authorization is given for a complete namespace an attacker could not only replace the existing values of the data elements but add further data elements in this namespace. Again, a verifier will validate the DeviceSignedItems with this falsified mdoc data.

To summarize: the security of DeviceSignedItems relies on the security of the mDL data elements and of the mdoc authentication private key in the secure area of the mDL, on top of the security of the issuing authority's keys. All of this can lead to an attacker being able to either fake the mDL of any arbitrary other holder, or even introduce fake mDLs.

An issuing authority can mitigate these DeviceSignedItems related risks by using secure areas for the mDL in line with security requirements as listed in 13.4.2, for example that are either issued by the issuing authority or the government itself, e.g. a physical driving licence or a national ID card with a secure chip. Alternative solutions exists whereby the secure area of the mDL will remain directly under control of the issuing authority, e.g. a hardware security module in the issuing authority infrastructure in case of an mDL for the use over the internet according to ISO/IEC TS 18013-7.

With respect to the support of IssuerSignedItems and DeviceSignedItems an issuing authority should therefore balance the privacy issues against the security risks. For the support of DeviceSignedItems the following recommendations apply:

- DeviceSignedItems should be used with the mdoc MAC Authentication, see 12.4.5, but not the mdoc ECDSA / EdDSA Authentication option, see 12.4.6, to counter the privacy risk of demonstrably authentic data.
- The issuing authority should consider reducing the security risk of DeviceSignedItems by using a combination of IssuerSignedItems and DeviceSignedItems; e.g. using IssuerSignedItems for person identifying data such as name, date of birth, and portrait, while using DeviceSignedItems only for (sensitive) non-identifying attributes such as driving privileges or certain domestic data elements. This prevents that an attacker can create mDLs for other identities if the mdoc authentication private key or the values of the data elements are compromised. However, it also limits the privacy advantages of using DeviceSignedItems.
- The issuing authority should provide information about the assurance level of the implementation and this information should be publicly accessible out-of-band to the verifier, in particular with respect to the implementation of the DeviceSignedItems.

E.7 PKI and trust framework

The integrity and authenticity of an mDL is protected through the use of cryptographic mechanisms and digital certificates managed by a Public Key Infrastructure (PKI) under control of the respective issuing authority (IA). At the core of this PKI is the issuing authority certificate authority (IACA), from which a specific set of certificates are generated and used to protect mDL data and transactions. The IACA (one per IA) is the root of trust for all mDL verifiers (or relying parties in general) to securely validate the integrity and authenticity of mDLs issued by the respective IA. It is therefore of utmost importance that IAs put strict security controls on the deployment, administration and operation of their IACA. A compromise of the IACA (e.g. disclosure of the IACA private key, fraudulent issuance of document signer certificates, etc.) can be exploited for forgery of mDLs and ultimately undermines the trust on the IA's mDLs. The issuing authority should ensure that adequate security measures are used to ensure a high level of security for the use of the document signer keys.

Dissemination of IACA certificates is also critical to a robust and secure ecosystem. Trusting a fake IACA opens room for forgery of mDLs, leading to possible attacks of impersonation, skimming, eavesdropping, amongst others. mDL verifiers should accept and trust an IACA root certificate through their own inspection, through a trusted VICAL provider or other means.

At domestic level, dissemination of the IACA root certificate is expected to be distributed by the IA through secured and well-known channels to the mDL verifiers (e.g. downloadable from a webpage on the TLS protected website of the IA).

However, for cross border and interoperable verification of mDLs, a more scalable and dynamic mechanism is necessary. In the absence of a worldwide organization that could operate a globally trusted directory of IACA root certificates, it is envisioned that some organizations (public or private) will collect the list of IACAs and make available that compilation to mDL verifiers, under the terms and conditions at their own discretion. Such organizations are referred to in this document as VICAL providers and the list of IACAs as VICAL.

Given the criticality of the information contained in the VICAL, a set of minimum security requirements are specified under Annex C. The requirements include provisions for the initial validation and due diligence procedures of IAs. The requirements also contain provisions for VICAL Providers to secure communication with IAs for the respective IACAs, data management and security controls over the VICAL and distribution of the VICAL to mDL verifiers.

When selecting VICAL Providers, mDL verifiers should assess the trustworthiness of the VICAL Provider amongst other criteria.

E.8 Auditability

E.8.1 Holder auditability

The mDL holder should be able to view an audit trail of their interactions with mDL readers. To accomplish this, mDLs should, at the mDL holder's choice, log information about each transaction in a form that can be reviewed by the mDL holder. The mDL should not make the transaction log available to anyone other than the mDL holder, except at the mDL holder's direction. The transaction log should include:

— the identity of the mDL verifier / mDL reader;

— which data elements were requested by the mDL reader;

— which data element requests were rejected by the mDL, due to mDL holder consent or other policy controls; and

— the date, time and location of the transaction.

To ensure that the audit log is not detrimental to mDL holder privacy, the mDL holder should have the capability to disable the audit log and delete logged transactions and mDL holder Authentication should be required to view it. The transaction log should be stored in secure mobile storage. The format of the transaction log is out of the scope of this document.

E.8.2 mDL verifier auditability

mDL verifiers should provide the mDL holder the ability to review data collected from their mDL transactions and to revoke consent to process or hold the mDL holder's data.

The mDL verifier should implement controls, which may include:

— register mDL readers individually and specify the names of the mDL verifier and each mDL reader operator if a common certificate is used;

— avoid logging mDL holder PII;

— log only the data required by law.

E.9 Anonymity and unlinkability

E.9.1 Applicability

This annex defines anonymity according to ISO/IEC 29100 as a "characteristic of information that does not permit a personally identifiable information principal to be identified directly or indirectly". It is noted that this explicitly includes required mitigations against de-anonymization techniques, which are defined as “any process in which anonymous data is cross-referenced with other sources of data to re-identify the anonymous data source” in ISO/IEC 20889.

Transactions between mDL holders and mDL verifiers should be anonymous and unlinkable under at least the following threat models.

— Tracking of identities by (potentially colluding) mDL verifiers: while mDL verifiers may be able to identify an mDL holder based on transmitted attributes in one transaction, a different transaction with the same mDL holder with a non-overlapping set of attributes should not be linkable to the previous transaction(s). Transactions without identifying attributes should remain anonymous to any Verifier.

— Linking pseudonyms or transactions across non-connected mDL verifiers: interactions of an mDL holder with different (domains of) mDL verifiers should not be linkable through cross-referencing the respective transaction logs.

— Tracking of identities by (global) passive adversaries: passive observation (eavesdropping) of local or remote network communication should not lead to de-anonymization or linking of transactions between mDL holders and Verifiers or the relationship between mDL holders and issuing authorities.

De-identification alone is not sufficient for anonymization, and unlinkability requires anonymization of metadata elements. Note that revocation techniques may compromise privacy guarantees retroactively. Systems that implement explicit revocation techniques beyond limiting document lifetimes should therefore provide backward unlinkability in addition to unlinkability. When an mDL is revoked by IA, the mDL is non-ambiguously targeted by revocation techniques that are not likely to comply with privacy principles for some aspects. The mDL holder to revoke is by definition identified and a revoked mDL can no longer be accepted and its bearer is recognized as such. Therefore, privacy established at transaction time after issuance should not be neglected at revocation time. Accordingly, backward unlinkability that is an important property that prevents the loss of anonymity of past signatures when a user is revoked, should be preserved. As example when an mDL is revoked, the signatures previously made with his device key should not become linkable if previously anonymous.

The following subsections provide an overview of technical and organizational mitigations to help provide anonymity and unlinkability.

E.9.2 Data minimization for metadata

It is important to note that anonymity or pseudonymity (e.g. through de-identification) on data attributes can be ineffective if metadata items are associated with multiple transactions. Common metadata elements that could lead to privacy compromise are hardware or network addresses, long-term public key material, or session tokens. Additionally, statistical traffic analysis can lead to re-identification attacks and therefore compromise anonymity through linking of transactions (even if the content of these transactions is confidential through end-to-end encryption in upper layers).

Therefore, data minimization techniques should be applied to both data and metadata. The following subsections describe approaches to minimizing exposure of long-term key material. Minimization of address elements depends on the respective transport (both in device engagement and data transfer phases).

— NFC: should not use static lower layer addresses.

— QR code: does not use lower layer addresses.

— Bluetooth® LE: LE privacy should be enabled including private reconnection addresses as of Bluetooth® 4.2 to randomize lower layer addresses.

— Wi-Fi Aware: Wi-Fi Aware device’s NAN Management Interface Address (NMI) used for discovery and NAN Data Interface Address(es) (NDIs) should be randomly generated every time a Wi-Fi Aware interface is enabled. In addition, Wi-Fi Aware interface unique identifiers such as NMI MAC addresses, UUIDs, and TCP port numbers should be changed at regular intervals, e.g. every 15 minutes, if possible, as recommended by the Wi-Fi Alliance Neighbor Awareness Networking Specification.

E.9.3 Unlinkability and forward secrecy through ephemeral identifiers

Ephemeral keys from mDL and mDL reader are used to establish an authenticated, encrypted secure channel without exposing any long-lived public keys to passive observers. Active man-in-the-middle attackers could see mDL and mDL reader authentication public keys. All ephemeral keys should be destroyed by both mDL and mDL reader after use to ensure forward secrecy against an eavesdropper who records the encrypted session and later compromises one or both.

E.9.4 Rotation of public keys

The possibility of linkability through the mDL Authentication keys should be taken into consideration by IAs when deciding policy for the frequency of mDL updates. Rotation of public keys during mDL updates will implement some resistance to tracking of mDL transactions by mDL verifiers. mDL updates can happen more frequently than mDL data is updated.

The mDL Authentication keys should be changed as often as possible so they do not become a unique identifier that can be used to link mDL transactions even when the exchanged data does not uniquely identify the mDL holder. The IA should implement solutions where the mDL authentication key is changed frequently. Triggers for authentication key rotation could be changes in the mDL data, number of days the key is active, or number of times the key is used.

mDLs may store a set of mDL authentication keys (together with a set of random numbers for all data elements and an MSO for each key) and choose an unused or random key for each transaction, or rotate keys for each transaction. In the event that all available mDL authentication keys have been used, the mDL may re-use a key, but re-use should be minimized and obtaining mDL holder consent should be considered.

E.9.5 MSO digests

An issuing authority should ensure that the number of digests in the MSO does not inadvertently leak any information. To prevent such a leakage an issuing authority can add digests that are not related to any data elements in the MSO and can use a fixed number of digests in all issued mDLs.

E.10 Tracking mDL usage

Issuing authorities and technology providers should not track mDL holders or the usage of any mDL. Auditability should be for each individual mDL holder and not permit issuing authorities or technology providers to track transactions, aggregate audit trails across mDL holders, manage mDL verifier to mDL holder relationships, prevent (censor) mDL holder to mDL verifier relationships, or otherwise facilitate usage of identity information in an mDL that is not strictly in the interest of the mDL holder.

For device retrieval, transaction information should not be logged or uploaded in any form that permits issuing authorities or technology providers to track usage of the mDL or the habits of mDL holders.

When using Status or Identifier lists, the Privacy Considerations in Token Status List (draft-ietf-oauth-status-list-12) should be observed by the issuer to prevent creating a tracking vector when a relying party checks if a certain MSO is revoked.

E.11 Collection limitation

mDL verifiers should not request all data elements, but should stick to requesting solely those data elements necessary for the transaction at hand.

During device retrieval nor the mDL not should respond with more data than was requested by the mDL verifier in order to fulfill the transaction. This document permits field-by-field retrieval of data elements. Consent should always be granted by the mDL holder with full knowledge of the requester, the requested data and purpose of granting consent.

E.12 Data minimization

E.12.1 Issuing authority data minimization

Issuing authorities should provision age verification statements to support mDL verifiers across the largest geographical scope that their mDL holders are expected to present their mDL. By provisioning ten age verification statements of age greater than 15 through 21, and 25, the current age and one more than the current age, mDL verifiers should be able to confirm age around the globe without revealing Date of Birth.

E.12.2 mDL verifier data minimization

mDL verifiers should not request all data elements, but should stick to requesting solely those data elements necessary for the transaction at hand. If a wine store, for instance, requires that the purchaser have government-issued ID that proves they are over-21 and compares the person to the portrait in order to verify identity, the mDL reader in this store can ask for age confirmation and portrait, and because the validation of signature on the data demonstrates a government-issued ID card, does not need to request additional data. They should not request the date of birth in order to calculate the over-21 value they require.

mDL verifiers should not retain or log information about the transaction without mDL holder consent. For instance, the wine store logs what is necessary to display compliance with regulations that require an employee to verify the purchaser is over 21 and their identity matches their photos, but should never log names, identifying numbers, and portraits of patrons.

E.13 mDL holder (user) authentication

Unlocking and use of the mDL should only be permitted with appropriate mDL holder authentication proving that the user is the intended mDL holder. This ensures that others cannot collect mDL data without the mDL holder.

User authentication may be required to unlock the device and the mDL. With this, it becomes more difficult for someone other than the mDL holder to access and unlock the mDL data or to present and use the mDL. However, many mobile device users enroll device-level authentication for shared users of the device that may not be the mDL holder (e.g. enrolling a child fingerprint to unlock a device in order to play games). mDL holder authentication should tie the present user to the mDL holder to whom the mDL was originally issued.

This document provides a method for identity verification performed by the mDL verifier by comparing the portrait received by the mDL reader to the person presenting the mDL. If this is not sufficient to meet the risk requirements of the transaction, the mDL reader may implement biometric comparison of the person presenting the mDL to the portrait.

Access to data elements and the authentication key should be guarded by mDL holder authentication. The authentication should be implemented in secure hardware as well.

E.14 mDL holder consent

E.14.1 Informed consent

No mDL data should be shared with any other party without informed consent. Informed Consent dictates that the mDL holder be given sufficient informed just-in-time notice about the data being requested, the entity requesting the data, and the purpose for the request. Informed Consent also requires consent gathering — that the mDL holder should be able to actively confirm or deny, given the informational notice, that their data is about to be shared with the mDL verifier. Consent Gathering can happen at two distinct times in the flow of a transaction.

In no case should a blanket authorization to share data with any mDL verifier or specific mDL verifiers be obtained from the mDL holder as part of Terms and Conditions, without a well explained pre-consent mechanism, or without a transaction-time informed consent process.

E.14.2 Operating system medical IDs

On mobile operating systems, there is functionality for the user to provide a “Medical ID”, including name and emergency contact. The user can opt in to provide access to the “Medical ID” by emergency responders from an unlocked phone. Issuing authorities should not provide access to mDL Data without mDL holder consent by any registered or unregistered terminal but can instead recommend users utilize the “Medical ID” feature.

E.14.3 Device engagement

The action of tapping or allowing a QR code to be scanned is proximal consent to connect the mDL to the mDL reader (device retrieval).

E.14.4 Data transfer

The mDL reader requests data and the mDL should obtain consent from the mDL holder to return the requested data elements and to validate the issuing authority signature on those data elements. This should be performed in one of two modes - pre-consent or transaction-time.

E.14.5 Pre-consent

Pre-consent enables mDL holders to configure specific mDL verifiers with whom they have a trust relationship to share data. Those mDL verifiers may be permitted access to a repeat set of mDL data without transaction-time consent. In all cases that pre-consent is long-lived, the mDL holder should clearly select this configuration and be aware that pre-consent is in place.

When pre-consent is used, it is recommended for the mDL holder to perform an intentional action at transaction time and/or for the mDL holder to be notified at transaction time that the mDL information was shared.

E.14.6 Transaction-time consent

Transaction-time consent is just-in-time informed notice and consent gathering during the processing of the request and before the response is provided to the mDL reader.

E.15 Provisioning accuracy and reliability

E.15.1 mDL provisioning

Achieving the privacy principle of accuracy, data quality, and integrity of the mDL across the mDL Ecosystem depends on accurate provisioning and reliability of mDL use. Nobody wants their identity documents in the hands of the wrong person, or to have someone else able to use their documents. Individual privacy and the trust of the ecosystem depend on accuracy.

In order to form trust an mDL should be provisioned accurately to the proper application on the mobile device of the intended mDL holder, and it should remain under the control of that intended mDL holder. Issuing authorities should select methods of provisioning an mDL to the mDL holder's mobile devices through accurate and secure methods that resist spoofing and impersonation.

NOTE Interfaces, protocols and services for provisioning the mDL are not within the scope of this document.

The issuing authority should ensure that the mDL be issued:

— according to Identity Assurance requirements of their jurisdiction (e.g. according to NIST SP 800-63 for Issuing Authorities in the United States);

— to the intended secure application that meets privacy and PII protection requirements;

— to the proper and intended holder of the mDL.

The issuing authority should ensure that the mDL:

— remains in control of that mDL holder throughout the lifetime of the credential;

— protects the credential and the mDL data;

— can only be used by the mDL holder or with the consent of the mDL holder;

— meets the mDL verifier's needs for Authentication Assurance particular to their region.

E.15.2 Provisioning to the right application

In the issuance process, there should be some means for the issuing authority to ensure the application can be identified as genuine and valid. Various proprietary techniques could apply to identify an application as a genuine. Various code obfuscation and tamper detection techniques are available off-the-shelf to facilitate hacking detection.

E.15.3 Provisioning to the right mDL holder

Many methods in use today for obtaining internet credentials – for example, possession of an email address, SMS of one-time passcode, and self-asserted enrollment – do not provide sufficient assurance on their own and will not result in trusted credentials or trust across the mDL ecosystem.

There are various techniques to facilitate issuing the mDL to the intended mDL holder:

— in-person provisioning;

— remote identity proofing;

— trusted referees or monitored remote proofing;

— user authentication from the App and mDL provisioning once authenticated;

— using multiple alternative personal channels (mail, SMS, email) to communicate a provisioning "trigger" that is combined to achieve identity assurance.

E.15.4 Continued use by the right user

The mDL should protect against unauthorized use of mDL by other than the mDL holder. This should include using authentication factors that tie the user to the mDL data and be wary of device authentication factors that support device-sharing (e.g. registering children's fingerprints for them to play games on a device). The mDL should not solely rely on device security.

The mDL should authenticate the mDL holder when launching the mDL to prevent against leaking mDL data to unauthorized users. At transaction-time, identity verification will be performed by the mDL verifier.

E.16 Data accuracy and freshness

Issuers sign the mDL data and therefore are declaring the accuracy of mDL data at the time of signing.

Issuers should establish policy for how often mDL data is to be refreshed. This policy is reflected in the mobile security object (MSO). mDL solutions should endeavor to refresh mDLs at least as often as policy states.

Verifiers should implement business decisions related to that freshness if mDL data is out of date.

E.17 Reader registration and authentication

mDL reader registration is an optional feature so that mDL solutions do not require registration of mDL readers to one central issuing authority in order to allow mDL transactions. Consolidating registrations as such would create a tracking vector as discussed above. Implementing mDL reader registration however allows various use cases when knowledge of the mDL reader is important, such as access to data elements that are restricted to authenticated mDL readers, providing informed consent to the user, and protecting against man-in-middle attacks that could compromise the confidentiality of mDL data.

It is worth noting that the standard ensures that access to mandatory mDL data elements as listed in Table 3 shall not be rejected for unregistered mDL readers.

If the terminal is registered to allow its authentication, the issuing authority should put in place policies and a registration process that guarantees the identity of the mDL reader and ensures the protection of the authentication secret and keys within the mDL reader.

When the mDL reader is authenticated, concern should be taken to make sure the element of its authentication does not become a unique identifier allowing to track the mDL reader itself. The mechanisms for rotation of public key can be applied to the mdoc reader authentication keys, as it is to the mDL keys, to achieve this.

While the mDL holder is likely to have a right to uniquely identify who initiated a data reading transaction from his mDL for later complaints, the terminal can be protected in order to avoid tracking from a third party observing the exchange, or from the mDL if it can initiate the transaction without active involvement of the mDL holder. A registry of the temporary authentication keys issued to mDL readers, managed by the issuing authority, can be the mechanism for this.

NOTE This document does not define a method by which access restriction to individual elements for specific mDL readers can be defined.

Annex F
(informative)

Reader identity certificate authority list (RICAL) provider

F.1 Overview

The decentralized Public Key Infrastructure (PKI) trust model used for mdoc verifier certification authority (CA) certificates requires a structured mechanism to distribute and manage the CA certificates relied upon by mdoc applications. For example, these CA certificates are the root and/or intermediate (subordinate) certificates of online and offline mdoc verifier certificates, and are used by mdoc applications to validate the X.509 certificate chain for reader authentication as defined in section 9.1.4 mdoc reader authentication.

Additionally, since there is no global organization overseeing all the disparate mdoc ecosystems, or assuming an operational role, it is not feasible to maintain a single, centralized repository containing all mdoc verifier CA certificates as a universal trust anchor for all mdoc participants.

To address this, a mechanism called Reader Identity Certificate Authority List (RICAL) is introduced, allowing an entity (RICAL Provider) to compile, operate, and offer such a trust anchor as a service to mdoc participants for specific ecosystems or domains.

The RICAL Provider outlines its service in a policy covering both technical and procedural controls. RICAL Providers can also expand the policy to include additional considerations (e.g., business, legal, etc.).

This annex does not define a specific policy or governance model for RICAL Providers, nor does it restrict the possibility of multiple providers operating simultaneously—whether from public or private entities, in competition or collaboration. Nevertheless, consistency in RICAL information across providers is expected.

Ultimately, RICAL is presented as one possible approach for establishing a secure and interoperable trust model for specific ecosystems and/or domains. It does not exclude other solutions, such as out of bound mechanisms or other mechanisms of trust lists.

The following is out of scope of this annex:

- How mdoc verifiers establish trust in the issuer of an mdoc (one possible solution is described in Annex C).
- How mdoc verifiers establish trust in the mdoc application.
- The definition of a reader CA certificate profile is out of scope for this version of the document. This annex may be updated in future revisions, based on implementation feedback, to define a certificate profile suitable for cross-ecosystem interoperability. For now, the profile is intentionally left undefined to preserve flexibility.

F.2 RICAL participants

F.2.1 Overview

This clause provides an overview of the RICAL participants.

The following is an organizational view of the RICAL participants.

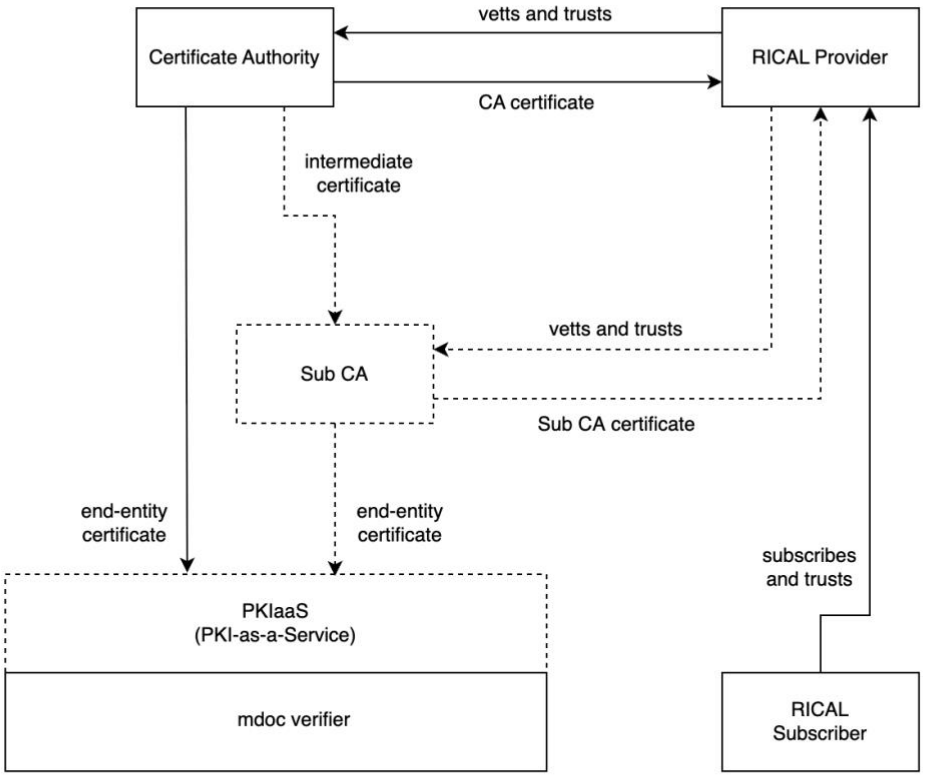

Figure F.1: Organizational RICAL View

The following is a technical view of the information flow between RICAL participants at transaction time.

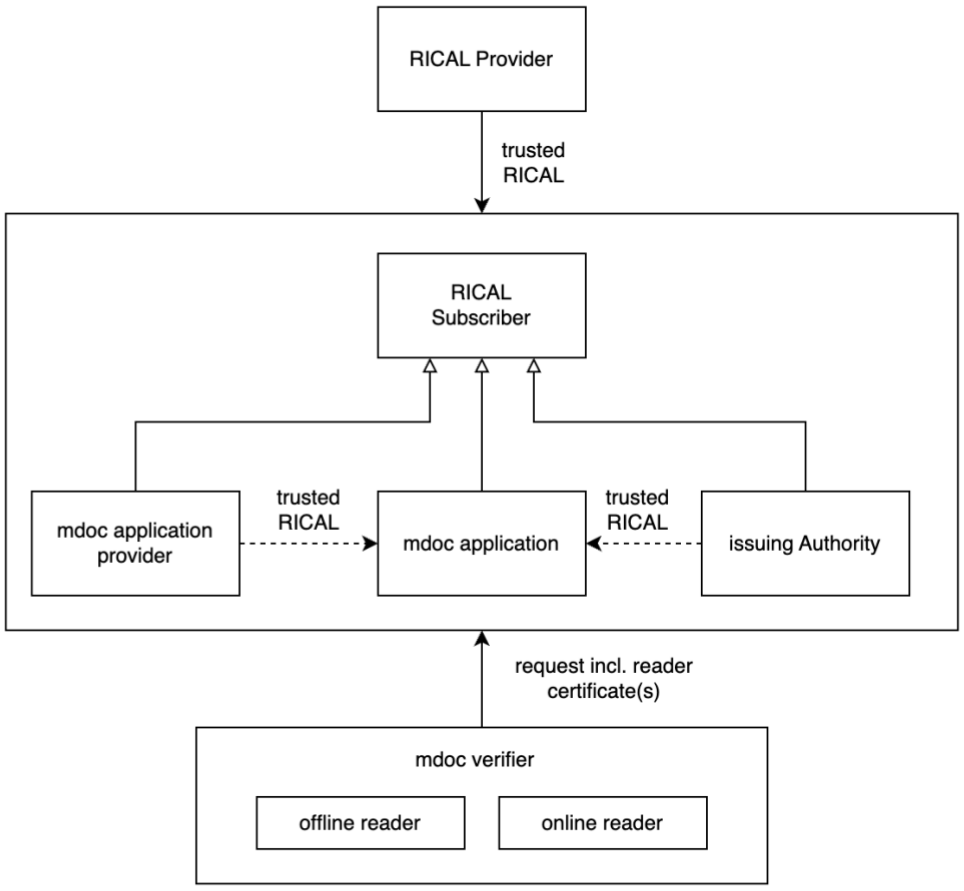

Figure F.2: Technical RICAL View

Note that in Figure G.2, black arrowheads indicate operational data flows directed toward the target entity, whereas white arrowheads signify that the target entity is a subtype or instance of the source entity, following an "is-a" relationship similar to class inheritance.

F.2.2 RICAL providers

RICAL providers are organizations responsible for delivering the RICAL, including activities such as:

- confirming the identity of the CA of mdoc verifiers;
- finding and regularly validating CAs’ point of contacts;
- ensure that a certificate authority complies with a set of requirements. The requirements can be set by the RICAL provider, or by other entities and enforced by the RICAL provider;
- collecting information from certificate authorities through the respective point of contacts;
- compiling the RICAL;
- creating and securing the RICAL according to the defined format;
- distributing the RICAL root certificate amongst RICAL Subscribers;
- distributing the RICAL amongst RICAL Subscribers;
- updating the RICAL regularly;
- defining and publishing a policy that states how trust is established, maintained, and relied upon.

Additionally, this annex does not prescribe specific methods for the generation, administration, or protection of key pairs. It is the responsibility of each RICAL Provider to ensure that key material is securely generated, managed, and safeguarded in accordance with appropriate security practices which is documented by their policy.

F.2.3 Certificate authorities

Certificate Authorities (CAs) are entities that issue X.509 certificates used to establish and maintain trust in digital communications and cryptographic operations. In the context of RICAL, CAs are responsible for issuing certificates to mdoc verifiers (i.e., relying parties), either directly or through authorized intermediaries. These certificates are used to authenticate mdoc reader applications during reader authentication flows.

Subordinate CAs (Sub CAs), also known as intermediate CAs, are CAs that have been issued a certificate by a higher-level (root or intermediate) CA. These Sub CAs are often used by large CAs, government trust frameworks, or PKIaaS providers to delegate certificate issuance responsibilities while maintaining separation of duties, scalability, and operational flexibility.

A CA included in a RICAL shall have the following purposes:

- To serve as trust anchor for verifier certificate validation;
- or to complete the certificate validation path;
- and to optionally assign Trust Constraints which define the conditions under which the CA's issued certificates may be trusted.

CAs included in a RICAL may be Sub CAs; therefore, Sub CAs may be designated as trust anchors.

CAs of Sub CAs may be omitted if the RICAL Provider does not intend to trust the entire hierarchy beneath those CAs. See F.3.2.3 for more information on restricting certificate usage.

A CA may include subsequent Sub CAs further down the certification path hierarchy if trust is extended appropriately and governance is verified by the RICAL Provider. In such cases, when mdoc verifier certificates and its use are validated (see F.3.2.6), trust constraints shall be applied at the final Sub CA level in the X.509 chain.

The RICAL Provider is not required to include the entire CA hierarchy. Instead, it explicitly may include only those CA certificates intended to serve as trust anchors for verifier certificate validation or to complete the certificate validation path, and enforcing Trust Constraints (see F.3.2.3). Additional Sub CA certificates required to complete the certification path may be provided by the mdoc verifier at transaction time.

The RICAL Provider shall not define a policy or RICAL extension to exclude specific Sub CAs.

F.2.4 RICAL subscribers

RICAL Subscribers receive the RICAL from the RICAL Provider and may redistribute it within the boundaries of the RICAL Subscriber, e.g., devices, installations. Typically, RICAL Subscribers are Issuing Authorities of credentials and/or mdoc application providers and/or mdoc applications.

A RICAL represents an assertion of trust by the RICAL Provider, and the RICAL Subscriber implicitly accepts the RICAL Provider's policy by using it—including the inclusion and exclusion criteria for CA certificates and any applied trust constraints. RICALs do not automatically confer trust in every environment and may be supplemented or overridden by other trust mechanisms.

RICAL root certificates serve as the ultimate trust anchor for RICAL Subscribers. This annex does not define methods or approaches for establishing trust in RICAL root certificates.

F.2.5 Mdoc verifiers

As defined in section 3.5, these can be organizations that provide online or proximity mdoc reader applications—such as websites or dedicated reader devices—and request credentials from mdoc applications that support or require reader authentication.

In practice, mdoc verifiers can obtain their certificates from Public Key Infrastructure as a Service (PKIaaS) providers. These service providers manage certificate issuance and lifecycle operations on behalf of them, and may act as registration authorities (RA) interfacing with trusted certificate authorities (CAs). PKIaaS offerings are particularly relevant for scalability and automation in ecosystems where many verifiers operate with similar assurance and policy requirements.

In the European context, mdoc verifiers can be registered through Relying Party Registrars managed by member states. These registrars issue access and/or registration certificates to relying parties. The RICAL would include the root CAs of these registrars for both registration and access certificates.

F.2.6 Mdoc application provider

mdoc Application Providers are entities responsible for developing, distributing, and maintaining mdoc applications that are used by holders to present mdocs. These providers ensure that mdoc applications support mdoc presentation protocols, user interface requirements, and integration mechanisms as specified by the relevant ecosystem or standard.

- An mdoc Application Provider plays a critical role in the trust chain by:
- Implementing support for verifying reader authentication and validation of the verifier certificate chain against one or more RICALs;
- Incorporating logic for interpreting RICAL-specific trust constraints;
- Ensuring secure communication and data handling practices during mdoc presentations;
- Maintaining and/or incorporating logic for up-to-date trust information based on the latest RICAL versions.

F.2.7 Mdoc application

An mdoc Application is a software component, typically running on a mobile device, that enables the holder to receive, store, and present mdocs as defined in section 3.2. The mdoc Application interacts with mdoc verifiers and participates in mdoc presentation protocols to establish secure sessions and support user consent-based data sharing.

- mdoc Applications play a critical role in the enforcement of trust relationships, including:
- Verifying the identity and trustworthiness of mdoc verifiers via reader authentication;
- Maintaining up-to-date trust information based on the latest RICAL versions;
- Validating verifier certificate chains against RICAL trust anchors and constraints;
- Applying additional policies or user-configured preferences to determine whether a verifier can be trusted;
- Ensuring user control over data disclosure and maintaining the integrity and confidentiality of the presented credentials.

The mdoc Application acts on behalf of the holder, who is the subject of the mdoc, i.e., digital credential. The application respects user privacy, facilitates informed consent, and ensures compliance with trust requirements defined in the ecosystem.

F.3 RICAL

F.3.1 Overview

The RICAL contains the CA certificates intended to be used by RICAL Subscribers as the trust anchor for the verification of authenticity and integrity of authentication assertions provided by mdoc verifiers during mdoc requests, e.g., reader authentication as defined in section 9.1.4 mdoc reader authentication.

To ensure authenticity, the RICAL is signed using the RICAL signer certificate, which is embedded within the structure itself. The corresponding RICAL root certificate (i.e., the trust anchor for verifying the RICAL signature) is distributed out of band to RICAL Subscribers.

In the absence of RICALs that include all existing CA certificates from mdoc verifiers of a certain ecosystem and/or domain, it may happen that mdoc applications may have to use and combine several RICALs to increase coverage.

Other additional mechanisms to obtain trust in mdoc verifiers may be used and combined with RICAL(s). Such are out of scope of this specification, and may include out of band mechanisms and other provisioning technologies.

F.3.2 RICAL structure

The RICAL is encapsulated and signed bythe untagged COSE_Sign1 structure as defined in RFC 9052. Within the `COSE_Sign1` structure, the `payload` shall be `RICAL`. The `external_aad` field used in the `Sig_structure` shall be a bytestring of size zero.

The alg element shall be present in the protected header of the COSE_Sign1. The RICAL signer certificate (i.e., the certificate corresponding to the private key used for signing) shall be included in the x5chain header as defined in RFC 9360, and it shall appear in the protected header. The RICAL root certificate shall not be included in the x5chain header, but additional intermediate certificates may be included if needed to complete the chain to the trust anchor.

The RICAL provider should use one of the following signature algorithms:

- ES256
- ES384
- ES512
- EdDSA

The curves used shall conform to those listed in Table 17.

F.3.2.1 RICAL Format

`RICAL` shall be CBOR encoded and formatted as follows:

```
RICAL = {
    "version" : tstr,   ; RICAL structure version, currently "1.0"
    "provider" : tstr,  ; Identifies the RICAL provider
    "date" : tdate                  ; date-time of RICAL issuance
    ? "nextUpdate" : tdate          ; next RICAL is expected to be issued before
this date-time
    ? "notAfter": tdate             ; date after which the RICAL is not valid
    "certificateInfos" : [*RICALCertificateInfo],    ; CA certificates included
in this RICAL
    ? "id" : uint,  ; Uniquely identifies the specific issue of the RICAL
    ? "latestRicalUrl" : tstr,          ; HTTPS URL of the latest RICAL
    ? "extensions" : Extensions,    ; Proprietary extensions
     "type" : tstr, ; Type of the RICAL
    * tstr => RFU   ; Reserved for future use (RFU)
}

Extensions = {
    * tstr => Ext   ; Proprietary extensions
}
```

The following fields are defined for the RICAL structure:

- version: Required. A string Indicating the RICAL version. The only defined value at present is "1.0".
- provider: Required. A string identifying the entity responsible for producing and signing the RICAL.
- issued: Required. Indicates when this RICAL was issued.
- nextUpdate: Optional. Suggests the latest expected issuance date of the next RICAL.
- notAfter: Optional. Indicates the RICAL is not valid after this time.
- certificateInfos: Required. An array of objects, each representing a CA certificate vetted and trusted by the RICAL Provider under their policy.
- id: Optional unique identifier for the issuance of the RICAL. If present, it shall monotonically increase between versions.
- latestRicalUrl: Optional. If present, shall be an HTTPS URL that can be used to fetch the most recent RICAL. The endpoint shall accept GET requests and return a CBOR payload with Content-Type application/cbor.
- type: Required. Collision-resistant identifier for the type of the RICAL. This annex defines the identifier org.iso.18013.5.1.reader_authentication for general-purpose RICALs that do not impose additional validation rules. RICAL Providers may define their own type identifiers along with associated validation, processing requirements and purposes. In such cases, the RICAL Provider shall publish this information in its policy.
- extensions: Optional field used for proprietary or future extensions.

All date-time fields (type `tdate`) shall be encoded in accordance with 13.4.2.

F.3.2.2 Certificate information

The following CDDL structure shall be used to represent a CertificateInfo used in a RicalInfo structure:

```
RICALCertificateInfo = {
    "certificate" : bstr,   ; DER-encoded X.509 certificate
    "serialNumber": biguint,    ; Serial number of the certificate
    "ski" : bstr,   ; Subject Key Identifier
    ? "type" : tstr,    ; Type of the certificate
    ? "trustContraints" : [+ TrustConstraint],  ; Conditions for certificate trust
    ? "name" : tstr,    ; Name of the CA
    ? "issuingCountry" : tstr,  ; ISO3166-1 or ISO3166-2 country of the CA
    ? "stateOrProvinceName" : tstr, ; State or province name of the CA
    ? "issuer" : bstr,  ; DER-encoded issuer field of the certificate (i.e. the complete
;Name structure)
    ? "subject" : bstr, ; DER-encoded subject field of the certificate (i.e. the complete
Name structure)
    ? "notBefore" : tdate,  ; Certificate validity start
    ? "notAfter" : tdate,   ; Certificate validity end
    ? "extensions" : Extensions,    ; Application-specific extensions
    * tstr => RFU   ; Reserved for future use (RFU)
}
```

The following fields are defined for the CertificateInfo structure:

- certificate: Required. The DER-encoded X.509 CA certificate.

- serialNumber: Required. The serial number of the CA certificate.

- isTrustAnchor: Required. A boolean value indicating whether the CA certificate is intended to be used as a trust anchor. If set to true, the certificate shall be treated as a trust anchor during certificate path validation. If set to false, the certificate shall not be used as a trust anchor.

- ski: Required. Subject Key Identifier of the CA certificate.

- aki: Optional. Authority Key Identifier of the issuer of the Sub CA. This field shall be present for Sub CAs.

- type: Optional. Collision-resistant identifier for the type of the certificate, e.g., urn:eu:rp:registration-certificate.

- trustConstraints: Optional. If present, it describes conditions under which end entity certificates under this CA certificate are trusted. See section G.3.1.4 Trust Constraints.

- name: Optional. String representing a human-readable name of the CA.

- issuingCountry: Optional. The ISO 3166-1 or ISO 3166-2 country code represents the country of the CA.

- stateOrProvinceName: Optional. The state or province name associated with the CA.

- issuer: Optional. The DER-encoded issuer field of the certificate (full X.500 Name).

- subject: Optional. The DER-encoded subject field of the certificate (full X.500 Name).

- notBefore: Optional. The date before which the certificate is not valid.

- notAfter: Optional. The date after which the certificate is not valid.

- extensions: Optional. An extension map for proprietary or future extensions.

F.3.2.3 Trust constraints

Trust Constraints define additional conditions under which end-entity certificates that chain to CAs listed in the RICAL are considered trusted. They are used to specify usage limitations or requirements that go beyond the inclusion of a CA certificate in the RICAL itself. Trust Constraints are particularly useful when the RICAL Provider's policy does not fully express, cannot verify, or cannot enforce such conditions at the time of RICAL issuance. These constraints allow RICAL Providers to express conditional trust while delegating enforcement to RICAL Subscribers.

It is the responsibility of the RICAL Subscriber (e.g., an mdoc application) to evaluate and enforce the applicable Trust Constraints during the certificate validation process as described in F.3.2.6.

This specification does not define any concrete Trust Constraints. It is expected that specific ecosystems or domains define and profile such constraints as needed, based on their operational, regulatory, or technical requirements.

Examples of such constraints can include:

- Restricting end-entity certificates to specific Extended Key Usage (EKU) values;
- Requiring the presence of one or more specific certificate policy OIDs;
- Enforcing recognized certificate profile URNs associated with a particular sector or use case;
- Limiting usage to certain document types (e.g., age verification) or purposes (e.g., offline or online verification, or login).

The following CDDL structure shall be used to represent a TrustConstraint used in a CertificateInfo structure:

```
TrustConstraint = {
    ? "policies": [+ tstr],
    ; Required policy OIDs in end-entity certificates
    ? "eku" : [+ tstr], ; List of allowed Extended Key Usage OIDs
    ? "pathLen" : uint, ; Maximum path length allowed from CA
    ? "certificateProfiles" : [+ tstr], ; Accepted URNs for certificate profiles
    ? "extensions" : Extensions ; Application-specific extensions
    * tstr => RFU   ; Reserved for future use (RFU)
}
```

F.3.2.4 RICAL processing

The validation of a RICAL involves verifying its authenticity, freshness, and structure. Adherence to defined Trust Constraints applies specifically during the validation of mdoc verifier end-entity certificates during mdoc presentment.

F.3.2.5 Validitation of the RICAL structure

A RICAL Subscriber shall validate a RICAL as follows:

- Signature Validation:
  - Obtain the RICAL root certificate (i.e., trust anchor) out of band.
  - Extract the RICAL signer certificate from the x5chain header of the COSE_Sign1 structure. Intermediate certificates may also be included in the x5chain to assist in path construction.
  - Validate the certificate chain of the RICAL signer certificate up to the RICAL root certificate. This includes certification path validation in accordance with RFC 5280, covering:
    - Signature chaining from the end-entity to a trust anchor;
    - Validity periods;
    - Name constraints and other X.509 extensions;
    - Revocation checking, if supported.

    - Verify the COSE_Sign1 signature using the validated signing certificate included in the x5chain header.
    - The external_aad input shall be a zero-length bstr.
    - Signature algorithms shall be limited to those specified in section 1.3.1 RICAL Format.
- Validity Period:
    - Ensure current time is within the bounds of RicalValidityInfo:
        - issued shall not be in the future.
    - If present, notAfter shall not be in the past.
    - If present, nextUpdate may be used as a signal to refresh the RICAL.
- Structure Integrity:
    - Ensure the RICAL itself is valid per section G.3.1 RICAL Format.
    - If a type element is included in the RicalInfo structure, additional type-specific validation rules may be applied.
    - The certificateInfos array shall not be empty.
    - Each included CA shall have a certificate validation path to a CertificateInfo element with isTrustAnchor set to true.

F.3.2.6 Validitation of the mdoc verifier certificate and its intended use

An mdoc application shall validate the end-entity certificate of an mdoc verifier that chains to a CA listed in the RICAL as follows:

- Ensure the RICAL itself is valid per the procedures defined in section G.3.2.1 Validation of the RICAL Structure.
- Certification Path Construction and Matching:
    - Identify matching CertificateInfo elements by comparing the certificate chain provided by the mdoc verifier, e.g., x5chain header in case of reader authentication, against CA certificates listed in the RICAL.
    - The CA certificate from the CertificateInfo element with isTrustAnchor set to true that is highest in the certificate validation path shall be used as the trust anchor.
    - Construct a certification path from the end-entity certificate to the identified CA certificates of the identified CertificateInfo elements listed in the RICAL.
    - Validate that the identified CA certificates and the end-entity certificate are well-formed, not expired, and not revoked (if revocation information is available).
    - The CertificateInfo element of the first certificate (bottom-up) in the certificate chain included in the RICAL shall be used to determine and apply the associated Trust Constraints. This certificate is not necessarily the root or trust anchor for the path validation, i.e., in case isTrustAnchor is false. Other CertificateInfo elements listed in the RICAL (e.g., Sub CAs) may still appear higher in the path. Only Trust Constraints of this first matching CertificateInfo shall be applied to the end-entity certificate.
- Validate the certification path in accordance with RFC 5280, including:
    - Signature chaining from the end-entity to a trust anchor;
    - Validity periods;
    - Name constraints and other X.509 extensions;
    - Revocation checking, if supported.
- Trust Constraint Evaluation:
    - If the first (bottom-up) identified CertificateInfo element includes a non-empty trustConstraints array element, the end-entity certificate or its usage shall satisfy at least one fully valid TrustConstraint

    - If a TrustConstraint includes an extensions field containing Trust Constraint types that are not understood or supported by the mdoc application, the entire TrustConstraint element shall be considered not satisfied.
    - If multiple TrustConstraint array elements are defined for a single CertificateInfo, each is evaluated independently.
    - The end-entity certificate and its intended usage are trusted only if at least one TrustConstraint is fully satisfied.
    - Trust Constraints from other CertificateInfo elements (even if also present in the path or RICAL) shall not be combined.
- Freshness Check (optional):
    - If latestRicalUrl is provided, a newer version of the RICAL may be fetched and the id or issued compared to assess freshness.

If a RICAL Subscriber uses RICALs from multiple RICAL Providers and encounters any conflicts — for example, in trust constraints or validity information — the resolution of such conflicts by the RICAL Subscriber is out of scope of this annex.

F.3.3 Handling multiple RICAL versions

When a new RICAL is issued, it may contain different or updated trust anchors, constraints, or other metadata. A RICAL Subscriber and/or mdoc application that has already validated a previous RICAL and has not yet fetched the latest version may continue to use the previously validated RICAL, provided it is still in accordance with section G.3.2.1 Validation of the RICAL Structure.

However, if a new RICAL is available and valid, the RICAL Subscriber and/or the mdoc application should use the latest version to ensure access to the most current and accurate trust information.

RICAL Subscribers and/or mdoc applications should implement periodic refreshing of the RICAL based on the `nextUpdate` field, if provided, and should support fallback behavior when the latest version cannot be retrieved.

If a RICAL Subscriber has multiple valid versions of a RICAL, it shall use the most recently fetched, i.e., downloaded, version. Newly fetched RICALs override previously retrieved versions.

RICAL Providers should revoke the RICAL signer certificate if a valid RICAL contains a CA that is later found to be compromised or no longer in compliance with the RICAL Provider's policy requirements. The RICAL Provider's policy shall define the conditions under which the RICAL signer certificate is revoked.

F.3.4 RICAL root certificate profile

There is currently no formally defined certificate profile for RICAL root certificates. However, implementers may refer to section B.1.2 IACA root certificate as a source of guidance. When doing so, it is important to consider that certain fields—such as CRL distribution points—are typically unnecessary or not applicable to RICAL root Certificates and should be omitted.

F.3.5 RICAL signer certificate profile

This certificate profile defines the RICAL signer certificates, establishing the minimum security parameters (key lengths, algorithms, policy IDs, etc.). CAs issuing certificates to RICAL Providers may choose to use equivalent or higher parameters, as well as other certificate fields and extensions that do not limit or reduce the overall security level.

The policy OID included in the RICAL signer certificate shall correspond to the RICAL Provider's documented policy, which defines technical, procedural, and—where applicable—legal and business requirements for operating as a RICAL Provider. This policy document may be referenced via a Certification Practice Statement (CPS) URI within the same certificatePolicies extension.

While this annex does not mandate a specific policy or governance model, it is essential that all RICAL Provider signer certificates include such a policy OID to ensure consistent interpretation, traceability, and interoperability across different ecosystems. The policy OID allows RICAL Subscribers and/or mdoc applications to determine whether a given RICAL instance meets their trust requirements based on known governance profiles.

Table F.1 — RICAL signer certificate profile

Certificate component	Section in RFC 5280	Presence	Criticality	Description
Version	4.1.2.1	M		Shall be `v3`.
Serial number	4.1.2.2	M		Non-sequential positive, non-zero integer, shall contain at least 63 bits of output from a CSPRNG, should contain at least 71 bits of output from a CSPRNG, maximum 20 octets.
Signature	4.1.2.3	M		Value shall match the OID in the signature algorithm (below).
Issuer	4.1.2.4	M		According to the Certification Authority issuing the RICAL signer certificate
Validity	4.1.2.5	M		
notBefore		M		Date on which the certificate validity period begins.
notAfter		M		Maximum of 1 187 days after "notBefore" date
Subject	4.1.2.6	M		Minimum required fields. Others may be present (e.g. Serial Number, State, Organization Unit, etc.).
Country (C)		M		Country code of jurisdiction of RICAL Provider. Encoded as `PrintableString`.
Organization (O)		M		Full registered name of the RICAL Provider. Encoded as `UTF8String`.
Common name (CN)		M		Name under which RICAL Provider operates RICAL service and is commonly known. Encoded as `UTF8String`.
Subject public key info	4.1.2.7	M		
algorithm		M		`1.2.840.10045.2.1` (Elliptic curve)
parameters		M		Implicitly specify curve parameters through an OID associated with one of the following curves specified in NIST SP 800-186: `1.2.840.10045.3.1.7` (Curve P-256) `1.3.132.0.34` (Curve P-384) `1.3.132.0.35` (Curve P-521) Or one of the following curves specified in RFC 5639: `1.3.36.3.3.2.8.1.1.7` (brainpoolP256r1) `1.3.36.3.3.2.8.1.1.9` (brainpoolP320r1) `1.3.36.3.3.2.8.1.1.11` (brainpoolP384r1) `1.3.36.3.3.2.8.1.1.13` (brainpoolP512r1)
subjectPublicKey		M		Public key shall be encoded in uncompressed form.
X.509v3 extensions	4.2	M		Further extensions may be present if they are marked non-critical.
Authority key identifier	4.2.1.1	M	NC	

Certificate component	Section in RFC 5280	Presence	Criticality	Description
keyIdentifier		M		Same value as the subject key identifier of the issuer's certificate
Subject key identifier	4.2.1.2	M	NC	SHA-1 hash of the subject public key `BIT STRING` value (excluding tag, length, and number of unused bits).
Key usage	4.2.1.3	M	C	
Digital signature				`0`
Non-repudiation				`1`
Key encipherment				`0`
Data encipherment				`0`
Key agreement				`0`
Key certificate signature				`0`
CRL signature				`0`
Encipher only				`0`
Decipher only				`0`
CRLDistributionPoints	4.2.1.13	M	NC	The 'reasons' and 'cRL Issuer' fields shall not be used.
distributionPoint		M		URI for CRL distribution point
Private internet extensions				
Authority information access	4.2.2.1	C	NC	Conditional, shall be present if the RICAL has an OCSP service.
Access description OCSP		M		
accessMethod		M		`1.3.6.1.5.5.7.48.1` (OCSP)
accessLocation		M		URI for corresponding OCSP service
Signature algorithm	4.1.1.2	M		Options: `1.2.840.10045.4.3.2` (ECDSA-with SHA256) `1.2.840.10045.4.3.3` (ECDSA-with SHA384) `1.2.840.10045.4.3.4` (ECDSA with SHA512)
Signature value	4.1.1.3	M		Value according to the signature algorithm. By creating this signature, the CA certifies the binding between the public key material and the subject of the certificate, i.e. the RICAL Provider.

Key

Presence:

M	mandatory
O	optional
C	conditional

Criticality:

C	critical
NC	not critical

Annex G
(informative)

IANA Considerations

G.1 COSE Elliptic Curves registration

This document defines the brainpool curves described in 9.1.5.2 for the IANA COSE Elliptic Curves registry defined in RFC 8152.

Name: brainpoolP256r1
Value: under consideration from IANA
Key Type: EC2
Description: BrainpoolP256r1
Change Controller: ISO/IEC JTC 001/SC 17/WG10
Reference: ISO/IEC 18013-5:2021, 9.1.5.2
Recommended: no

Name: brainpoolP320r1
Value: under consideration from IANA
Key Type: EC2
Description: BrainpoolP320r1
Change Controller: ISO/IEC JTC 001/SC 17/WG10
Reference: ISO/IEC 18013-5:2021, 9.1.5.2
Recommended: no

Name: brainpoolP384r1
Value: under consideration from IANA
Key Type: EC2
Description: BrainpoolP384r1
Change Controller: ISO/IEC JTC 001/SC 17/WG10
Reference: ISO/IEC 18013-5:2021, 9.1.5.2
Recommended: no

Name: brainpoolP512r1
Value: under consideration from IANA
Key Type: EC2
Description: BrainpoolP512r1
Change Controller: ISO/IEC JTC 001/SC 17/WG10
Reference: ISO/IEC 18013-5:2021, 9.1.5.2
Recommended: no

G.2 URI registration

This document defines the “mdoc” URI defined in 8.2.2.3 for the IANA Uniform Resourse Identifier (URI) Schemes registry defined in RFC 7595.

URI scheme name: mdoc
Status: permanent
Applications/protocols that use this scheme name: the mdoc URI scheme is inteded to be used by mobile document applications
Contact: [To be filled in by ISO secretariat]

Change controller: ISO/IEC JTC 001/SC 17/WG10
Reference: ISO/IEC 18013-5:2021

G.3 CWT status mechanisms registry

This document defines the identifier list status mechanism described in 12.3.6 for the IANA CWT status mechanisms registry.

Status mechanism value: identifier_list

Status mechanism description: A token status list, containing up-to-date status information on multiple tokens where the tokens are identified using an identifier.

Change controller: ISO/IEC JTC 1 / SC 17 / WG10.

Specification document: 12.3.6 of this document.

Annex H
(informative)

Structures overview

H.1 Overview

This section contains an informative overview of the CDDL of the structures as defined in this document.

```
; INTERNAL, Data Elements
; ------------------------------------------------------------------------------
DocType = tstr
NameSpace = tstr
DataElementIdentifier = tstr
DataElementValue = any

RFU = any
Ext = any
Version = tstr

; INTERNAL, Security structures
; ------------------------------------------------------------------------------
SessionTranscript = [
    DeviceEngagementBytes / null,
    EReaderKeyBytes / null,
    Handover
]

DeviceEngagementBytes = #6.24(bstr .cbor DeviceEngagement)

Handover = QRHandover / NFCHandover / Ext

QRHandover= null

NFCHandover = [
    bstr        ; Binary value of the Handover Select Message
    bstr / null ; Binary value of the Handover Request Message,
                ; shall be null if NFC Static Handover was used
]

SessionTranscriptBytes = #6.24(bstr .cbor SessionTranscript)

; INTERNAL, Device Engagement
; ------------------------------------------------------------------------------
DeviceEngagement =
{
    0: Version,
    1: Security,
    ? 2: DeviceRetrievalMethods,
    ? 5: OriginInfos,
    ? 6: Capabilities,
    * uint => RFU,
    * nint => Ext
}
```

```
Security = [
    int,    ; Cipher suite identifier
    EDeviceKeyBytes
]

DeviceRetrievalMethods = [
    + DeviceRetrievalMethod
]

DeviceRetrievalMethod = [
    uint,   ; Type
    uint,   ; Version
    RetrievalOptions
]

OriginInfos = [
    * OriginInfo
]

OriginInfo = RFU

Capabilities = {
    ? 2: HandoverSessionEstablishmentSupport,
    ? 3: ReaderAuthAllSupport
    * int => RFU
}

HandoverSessionEstablishmentSupport = bool
ReaderAuthAllSupport = bool


RetrievalOptions = WifiOptions / BleOptions / NfcOptions / RFU



; INTERNAL, Device Engagement RetrievalOptions
; ---------------------------------------------------------------------------
WifiOptions = {
    ? 0: tstr,      ; Pass-phrase Info Pass-phrase
    ? 1: uint,      ; Channel Info Operating Class
    ? 2: uint,      ; Channel Info Channel Number
    ? 3: bstr       ; Band Info Supported Bands
    * uint => RFU,
}

BleOptions = {
    0 : bool,       ; Indicates support for mdoc peripheral server mode
    1 : bool,       ; Indicates support for mdoc central client mode
    ? 10 : bstr,    ; UUID for mdoc peripheral server mode
    ? 11 : bstr     ; UUID for mdoc client central mode
    ? 20 : bstr     ; mdoc BLE Device Address for mdoc peripheral server mode
    * uint => RFU,
}

NfcOptions = {
    0 : uint,       ; Maximum length of command data field
    1 : uint        ; Maximum length of response data field
    * uint => RFU,
}

; INTERNAL, Reader Engagement
; ------------------------------------------------------------------------------
ReaderEngagement =
{
    0: tstr,            ; Version
    * uint => RFU,
    * nint => Ext
}

; INTERNAL, mdoc BLE Service Data
; ------------------------------------------------------------------------------


MdocBleServiceData = {
    ? 0 : MdocBlePsm,
    * int => RFU
}

MdocBlePsm = int

; INTERNAL, Device Request
; ------------------------------------------------------------------------------

DeviceRequest = {
    "version" : Version,
    "docRequests" : [+ DocRequest],
    ? "deviceRequestInfo" : DeviceRequestInfoBytes,
    ? "readerAuthAll" : [+ReaderAuthAll],
    * tstr => RFU
}

DeviceRequestInfoBytes = #6.24(bstr .cbor DeviceRequestInfo)

DocRequest = {
    "itemsRequest" : ItemsRequestBytes,
    ? "readerAuth" : ReaderAuth,
    * tstr => RFU
}

ItemsRequestBytes = #6.24(bstr .cbor ItemsRequest)

ItemsRequest = {
    "docType" : DocType,
    "nameSpaces" : NameSpaces,
    ? "requestInfo" : DocRequestInfo,
    * tstr => RFU
}

NameSpaces = {
    + NameSpace => DataElements
}

DataElements = {
    + DataElementIdentifier => IntentToRetain
}
```

```
IntentToRetain = bool

DocRequestInfo = {
    ? "alternativeDataElements": [+ AlternativeDataElementsSet]
    ? "issuerIdentifiers": [+IssuerIdentifier],
    ? "uniqueDocSetRequired": UniqueDocSetRequired,
    ? "maximumResponseSize": MaximumResponseSize,
    ? "zkRequest": ZkRequest,
    * tstr => Ext
}

IssuerIdentifier = bstr
UniqueDocSetRequired = bool
MaximumResponseSize = uint

AlternativeDataElementsSet = {
    "requestedElement": ElementReference,
    "alternativeElementSets": [+AlternativeElementSet],
    * tstr => RFU
}

AlternativeElementSet = [+ ElementReference]

ElementReference = [NameSpace, DataElementIdentifier]


DeviceRequestInfo = {
    ? "useCases": [+UseCase],
    * tstr => Ext
}

UseCase = {
    "mandatory" : bool,
    "documentSets" : [+ DocumentSet],
    ? "purposeHints" : PurposeHints,
    * tstr => RFU
}

DocumentSet = [+ DocRequestID]
DocRequestID = uint

PurposeHints = {
    + PurposeControllerId => PurposeCode
}

PurposeControllerId = tstr
PurposeCode = int

ZkRequest = {
    "systemSpecs": [+ ZkSystemSpec],
    "zkRequired": ZkRequired,
    * tstr => RFU
}

ZkRequired = bool

ZkSystemSpec = {
    "zkSystemId": ZkSystemId
    "system": ZkSystem,
    "params": ZkParams,
    * tstr => RFU
}

ZkSystem = tstr
ZkSystemId = tstr
ZkParams = { * tstr => Ext}



; INTERNAL, Device Response
; ------------------------------------------------------------------------------

DeviceResponse = {
    "version" : tstr,
    ? "documents" : [+Document],
    ? "zkDocuments" : [+ ZkDocument],
    ? "documentErrors": [+DocumentError],
    "status" : uint,
    * tstr => RFU
}

Document = {
    "docType" : DocType,
    "issuerSigned" : IssuerSigned,
    "deviceSigned" : DeviceSigned,
    ? "errors" : Errors,
    * tstr => RFU
}

DocumentError = {
    DocType => ErrorCode
}

IssuerSigned = {
    "issuerAuth" : IssuerAuth,
    ? "nameSpaces" : IssuerNameSpaces,
    * tstr => RFU
}


IssuerNameSpaces = {
    + NameSpace => [ + IssuerSignedItemBytes ]
}

IssuerSignedItemBytes = #6.24(bstr .cbor IssuerSignedItem)

IssuerSignedItem = {
    "digestID" : uint,
    "random" : bstr,
    "elementIdentifier" : DataElementIdentifier,
    "elementValue" : DataElementValue
    * tstr => RFU
}

DeviceSigned = {
    "nameSpaces" : DeviceNameSpacesBytes,
    "deviceAuth" : DeviceAuth,
    * tstr => RFU
}

DeviceNameSpacesBytes = #6.24(bstr .cbor DeviceNameSpaces)

DeviceNameSpaces = {
    * NameSpace => DeviceSignedItems
}

DeviceSignedItems = {
    + DataElementIdentifier => DataElementValue
}

DeviceAuth = {
    "deviceSignature" : DeviceSignature //     ; "//" means OR
    "deviceMac" : DeviceMac
    * tstr => RFU
}

Errors = {
    + NameSpace => ErrorItems
}

ErrorItems = {
    + DataElementIdentifier => ErrorCode
}

ErrorCode = int


ZkDocument = {
    "documentData": ZkDocumentDataBytes,
    "proof": bstr,
    * tstr => RFU
}

ZkDocumentDataBytes = #6.24(bstr .cbor ZkDocumentData)

ZkDocumentData = {
    "docType": DocType,
    "zkSystemId": ZkSystemId,
    "timestamp": full-date,
    ? "issuerSigned" : ZkNameSpaces,
    ? "deviceSigned" : ZkNameSpaces,
    ? "msoX5chain": COSE_X509,
    * tstr => RFU
}

ZkNameSpaces = {
    + NameSpace => [ + ZkSignedItem ]
}

ZkSignedItem = {
    "elementIdentifier": DataElementIdentifier,
    "elementValue": DataElementValue,
    * tstr => RFU
```

```
}



; INTERNAL, DrivingPrivileges
; ------------------------------------------------------------------------------
DrivingPrivileges = [
    * DrivingPrivilege
]

DrivingPrivilege = {
    "vehicle_category_code" : tstr      ; Vehicle category code as per ISO/IEC 18013-1 Annex
B
    ? "issue_date" : full-date           ; Date of issue encoded as full-date
    ? "expiry_date" : full-date          ; Date of expiry encoded as full-date
    ? "codes" : [+Code]                  ; Array of code info
    * tstr => RFU
}

Code = {
    "code": tstr                         ; Code as per ISO/IEC 18013-2 Annex A
    ? "sign": tstr                       ; Sign as per ISO/IEC 18013-2 Annex A
    ? "value": tstr                      ; Value as per ISO/IEC 18013-2 Annex A
    * tstr => RFU
}

; INTERNAL, SessionEstablishment
; ------------------------------------------------------------------------------
SessionEstablishment = {
     "eReaderKey" : EReaderKeyBytes,
     "data" : bstr                    ; Encrypted mdoc request
    * tstr => RFU
}

; INTERNAL, SessionData
; ------------------------------------------------------------------------------
SessionData = {
    ? "data" : bstr         ; Encrypted mdoc response or mdoc request
    ? "status" : uint       ; Status code
    * tstr => RFU
}

; INTERNAL, EDeviceKey and EReaderKey
; ------------------------------------------------------------------------------
EDeviceKey = COSE_Key                          ; Containing EDeviceKey.Pub
EReaderKey = COSE_Key                          ; Containing EReaderKey.Pub
EDeviceKeyBytes = #6.24(bstr .cbor EDeviceKey)
EReaderKeyBytes = #6.24(bstr .cbor EReaderKey)


; INTERNAL, MobileSecurityObject
; ------------------------------------------------------------------------------
IssuerAuth = COSE_Sign1     ; The payload is MobileSecurityObjectBytes

MobileSecurityObjectBytes = #6.24(bstr .cbor MobileSecurityObject)
```

```
MobileSecurityObject = {
    "version" : tstr,
    "digestAlgorithm" : tstr,
    "valueDigests" : ValueDigests,
    "deviceKeyInfo" : DeviceKeyInfo,
    "docType" : tstr,
    "validityInfo" : ValidityInfo
    ? "status": Status
    * tstr => RFU
}

DeviceKeyInfo = {
    "deviceKey" : DeviceKey
    ? "keyAuthorizations" : KeyAuthorizations,
    ? "keyInfo" : KeyInfo
    * tstr => RFU
}

DeviceKey = COSE_Key

KeyAuthorizations = {
    ? "nameSpaces" : AuthorizedNameSpaces
    ? "dataElements" : AuthorizedDataElements
    * tstr => RFU
}

AuthorizedNameSpaces = [+ NameSpace]
AuthorizedDataElements = {+ NameSpace => DataElementsArray}
DataElementsArray = [+ DataElementIdentifier]

KeyInfo = {
    * uint => RFU,
    * nint => Ext
}

ValueDigests = {
    + NameSpace => DigestIDs
}

DigestIDs = {
    + DigestID => Digest
}

ValidityInfo = {
    "signed" : tdate,
    "validFrom" : tdate,
    "validUntil" : tdate,
    ? "expectedUpdate" : tdate
    * tstr => RFU
}

Status = {
    ? "identifier_list" : IdentifierListInfo,
    ? "status_list" : StatusListInfo,
    * tstr => RFU
}
```

```
DigestID = uint
Digest = bstr

; INTERNAL, IdentifierList
; ------------------------------------------------------------------------
IdentifierListInfo = {
    "id" : Identifier,
    "uri": URI,
    ? "certificate": Certificate
    * tstr => RFU
}

IdentifierList = {
    "identifiers" : { * Identifier => IdentifierInfo },
    ? "aggregation_uri" : Aggregation_uri
    * tstr => RFU
}

IdentifierInfo = {}
Identifier = bstr
Aggregation_uri = tstr


URI = tstr
Certificate = bstr


; INTERNAL, DeviceAuthentication
; ------------------------------------------------------------------------
; For DeviceMac and DeviceSignature, use a null value for the payload.
; The detached content is DeviceAuthenticationBytes
DeviceMac = COSE_Mac0
DeviceSignature = COSE_Sign1

DeviceAuthenticationBytes = #6.24(bstr .cbor DeviceAuthentication)

DeviceAuthentication = [
    "DeviceAuthentication",
    SessionTranscript,
    DocType,
    DeviceNameSpacesBytes
]




; INTERNAL, ReaderAuthentication
; ------------------------------------------------------------------------
; For ReaderAuth, use a null value for the payload.
; The detached content is ReaderAuthenticationBytes
ReaderAuth = COSE_Sign1

ReaderAuthenticationBytes = #6.24(bstr .cbor ReaderAuthentication)

ReaderAuthentication = [
    "ReaderAuthentication",
    SessionTranscript,
    ItemsRequestBytes
]

; For ReaderAuthAll, use a null value for the payload.
; The detached content is ReaderAuthenticationBytes
ReaderAuthAll = COSE_Sign1

ReaderAuthenticationAllBytes = #6.24(bstr .cbor ReaderAuthenticationAll)

ReaderAuthenticationAll = [
    "ReaderAuthenticationAll",
    SessionTranscript,
    ItemsRequestBytesAll,           ; Order as in the mdoc request
    DeviceRequestInfoBytes / null
]

ItemsRequestBytesAll = [*ItemsRequestBytes]



; INTERNAL, VICAL
; ------------------------------------------------------------------------------
VICAL = {
    "version" : Version             ; VICAL structure version, currently "1.0"
    "vicalProvider" : tstr          ; Identifies the VICAL provider
    ? "vicalIssueID" : uint          ; identifies the specific issue of the VICAL, shall be
unique and monotonically increasing
    "date" : tdate                  ; date-time of VICAL issuance
    ? "nextUpdate" : tdate          ; next VICAL is expected to be issued before this date-
time
    ? "notAfter": tdate             ; date after which the VICAL is not valid
    "certificateInfos" : [*CertificateInfo]
    ? "extensions" : Extensions     ; Can be used for proprietary extensions
    ? "vicalURI" : tstr             ; URI where this VICAL can be retrieved
    * tstr => RFU                   ; To be used for future extensions, all values are RFU
}

CertificateInfo = {
    "certificate" : bstr            ; DER-encoded X.509 certificate
    "serialNumber" : biguint        ; value of the  serial number field of the certificate
    "ski" : bstr                       ; value of the Subject Key Identifier field of the
certificate
    "docType" : [+ DocType]           ; DocType for which the certificate may be used as a
trust point
    ? "certificateProfile" : [+ CertificateProfile] ; Type of certificate
    ? "issuingAuthority" : tstr     ; Name of the certificate issuing authority
    ? "issuingCountry" : tstr         ; ISO3166-1 or ISO3166-2 depending on the issuing
authority
    ? "stateOrProvinceName" : tstr   ; State or province name of the certificate issuing
authority
    ? "issuer" : bstr               ; DER-encoded Issuer field of the certificate (i.e. the
complete Name structure)
    ? "subject" : bstr                ; DER-encoded Subject field of the certificate (i.e.
the complete Name structure)
    ? "notBefore" : tdate           ; value of the notBefore field of the certificate
    ? "notAfter" : tdate            ; value of the notAfter field of the certificate
    ? "extensions" : Extensions     ; Can be used for proprietary extensions
    * tstr => RFU                   ; To be used for future extensions, all values are RFU
}
```

```
Extensions = {* tstr => Ext}        ; Can be used for proprietary extensions

CertificateProfile = tstr           ; Uniform Resource Name (URN) according to RFC 8141

; INTERNAL, Full-date
; ---------------------------------------------------------------------------
full-date = #6.1004(tstr)               ; Tag 1004 as per RFC 8943


; INTERNAL, RICAL
; ---------------------------------------------------------------------------


RICAL = {
    "version" : tstr,   ; RICAL structure version, currently "1.0"
    "provider" : tstr,  ; Identifies the RICAL provider
    "date" : tdate                  ; date-time of RICAL issuance
    ? "nextUpdate" : tdate          ; next RICAL is expected to be issued before this date-
time
    ? "notAfter": tdate             ; date after which the RICAL is not valid
    "certificateInfos" : [*RICALCertificateInfo],     ; CA certificates included in this
RICAL
    ? "id" : uint,  ; Uniquely identifies the specific issue of the RICAL
    ? "latestRicalUrl" : tstr,          ; HTTPS URL of the latest RICAL
    ? "extensions" : Extensions,    ; application-specific extensions
     "type" : tstr, ; Type of the RICAL
    * tstr => RFU   ; Reserved for future use (RFU)
}

;Extensions = {
;    * tstr => Ext   ; Proprietary extensions
;}




RICALCertificateInfo = {
    "certificate" : bstr,   ; DER-encoded X.509 certificate
    "serialNumber": biguint,    ; Serial number of the certificate
    "ski" : bstr,   ; Subject Key Identifier
    ? "type" : tstr,    ; Type of the certificate
    ? "trustContraints" : [+ TrustConstraint],  ; Conditions for certificate trust
    ? "name" : tstr,    ; Name of the CA
    ? "issuingCountry" : tstr,  ; ISO3166-1 or ISO3166-2 country of the CA
    ? "stateOrProvinceName" : tstr, ; State or province name of the CA
    ? "issuer" : bstr,  ; DER-encoded issuer field of the certificate (i.e. the complete
;Name structure)
    ? "subject" : bstr, ; DER-encoded subject field of the certificate (i.e. the complete
Name structure)
    ? "notBefore" : tdate,  ; Certificate validity start
    ? "notAfter" : tdate,   ; Certificate validity end
    ? "extensions" : Extensions,    ; application-specific extensions
    * tstr => RFU   ; Reserved for future use (RFU)
}

TrustConstraint = {
    ? "policies": [+ tstr],
    ; Required policy OIDs in end-entity certificates
    ? "eku" : [+ tstr], ; List of allowed Extended Key Usage OIDs
    ? "pathLen" : uint, ; Maximum path length allowed from CA
    ? "certificateProfiles" : [+ tstr], ; Accepted URNs for certificate profiles
    ? "extensions" : Extensions ; Application-specific extensions
    * tstr => RFU   ; Reserved for future use (RFU)
}
```

Bibliography

[1] ISO/IEC 14443-3:2018, *Cards and security devices for personal identification — Contactless proximity objects — Part 3: Initialization and anticollision*

[2] ISO/IEC 15408 (all parts), *Information security, cybersecurity and privacy protection — Evaluation criteria for IT security*

[3] ISO/IEC 17021-1, *Conformity assessment — Requirements for bodies providing audit and certification of management systems — Part 1: Requirements*

[4] ISO/IEC 18013-3, *Information technology — Personal identification — ISO-compliant driving licence — Part 3: Access control, authentication and integrity validation*

[5] ISO/IEC 19790, *Information technology — Security techniques — Security requirements for cryptographic modules*

[6] ISO/IEC 20889, *Privacy enhancing data de-identification terminology and classification of techniques*

[7] ISO/IEC 27006 (all parts), *Requirements for bodies providing audit and certification of information security*

[8] ISO/IEC 29100, *Information technology — Security techniques — Privacy framework*

[9] AAMVA Mobile Driver's License Functional Needs White Paper version 0.9

[10] Bluetooth SIG, *Bluetooth Core Specification, Version 5.2, December 2019*

[11] Bluetooth SIG, *Supplement to the Bluetooth Core Specification, Revision v9, December 2019*

[12] FIPS 140-2, *Security Requirements for Cryptographic Modules*

[13] NFC Forum, *Bluetooth Secure Simple Pairing Using NFC, Version 1.2, May 2019*

[14] NFC Forum, *Data Exchange Format (NDEF) Technical Specification, Version 1.0*

[15] NFC Forum, *Type 4 Tag Version 1.1*

[16] NIST SP 800-63, *Digital Identity Guidelines*

[17] NIST SP 800-157, *H. Ferraiolo et al., Guidelines for Derived Personal Identity Verification (PIV) Credentials, December 2014*

[18] OpenID Connect Dynamic Client Registration 1.0, *N. Sakimura et al, November 2014*

[19] RFC 3279, *W.Polk et al, Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, April 2002*

[20] RFC 3339, *G. Klyne et al., Date and Time on the Internet: Timestamps, July 2002*

[21] RFC 5480, *S. Turner et al., Elliptic Curve Cryptography Subject Public Key Information, March 2009*

[22] RFC 6960, *S. Santesson et al., X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, June 2013*

[23] RFC 7231, *R. Fielding et al., Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*

[24] RFC 8446, *E. Rescorla et al., The Transport Layer Security (TLS) Protocol Version 1.3, August 2018*

[25] RFC 8610, *H. Birkholz et al., Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures, June 2019*

[26] UN CONVENTION ON ROAD TRAFFIC. Vienna, 8 November 1968, as amended by Amendment of UN Convention on Road Traffic, Vienna of 29 March 2011

[27] UN CONVENTION ON ROAD TRAFFIC. Geneva, 19 September 1949