ETSI TS 119 152-1 V1.1.1 (2026-03)

TECHNICAL SPECIFICATION

Electronic Signatures and Trust Infrastructures (ESI); CB AdES (CBOR-AdES) digital signatures; Part 1: Building blocks and CB-AdES baseline signatures

Reference
DTS/ESI-0019152-1

Keywords
CB-AdES, CBOR, electronic signature

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Electronic Signatures and Trust Infrastructures (ESI).

The present document is part 1 of a multi-part deliverable covering CB-AdES digital signatures, as identified below:

Part 1: **"Building blocks and CB-AdES baseline signatures";**

Part 2: "Extended CB-AdES signatures".

NOTE: At the time of publication of the present document, part 2 does not exist.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

Electronic commerce has emerged as a frequent way of doing business between companies across local, wide area and global networks. Trust in this way of doing business is essential for the success and continued development of electronic commerce. It is therefore important that companies using this electronic means of doing business have suitable security controls and mechanisms in place to protect their transactions and to ensure trust and confidence with their business partners. In this respect digital signatures are an important security component that can be used to protect information and provide trust in electronic business.

The present document is intended to cover digital signatures supported by PKI and public key certificates, and aims to meet the general requirements of the international community to provide trust and confidence in electronic transactions, including, amongst other, applicable requirements from Regulation (EU) No 2024/1183 [i.1].

The present document can be used for any transaction between an individual and a company, between two companies, between an individual and a governmental body, etc. The present document is independent of any environment. It can be applied to any environment e.g. smart cards, SIM cards, special programs for electronic signatures, etc.

The present document is part of a rationalized framework of standards (see ETSI TR 119 000 [i.4]).

1 Scope

The present document:

1) Specifies a CBOR [1] format for AdES signatures (CB-AdES signatures hereinafter) built on CBOR Object Signing and Encryption (COSE hereinafter) as specified in IETF RFC 9052 [2]. For this, the present document:

   - Extends the CBOR Object Signing and Encryption specified in IETF RFC 9052 [2] by defining an additional set of CBOR header parameters that can be incorporated in the COSE Headers (either in its protected headers map or its unprotected headers map). Many of these new header parameters have the same semantics as the attributes/properties defined in CAdES [i.2], XAdES [8], and JAdES [9] digital signatures. Other header parameters are defined to meet specific requirements that current COSE cannot meet (e.g. for explicitly referencing detached payload). These new header parameters and their corresponding types are defined in CDDL, which is specified in IETF RFC 8610 [5].

   - Specifies the mechanisms for incorporating the aforementioned CBOR components in COSE [2] to build CB-AdES signatures, offering the same features as CAdES, XAdES, and JAdES in CBOR syntax, and therefore fulfilling the same requirements (such as the long-term validity of digital signatures).

2) Defines four levels of CB-AdES baseline signatures addressing incremental requirements to maintain the validity of the signatures over the long term. Each level requires the presence of certain CB-AdES header parameters, suitably profiled for reducing the optionality as much as possible. The aforementioned levels provide the basic features necessary for a wide range of business and governmental use cases for electronic procedures and communications to be applicable to a wide range of communities when there is a clear need for interoperability of digital signatures used in electronic documents.

NOTE 1: ETSI EN 319 102-1 [i.3] specifies procedures for creation, augmentation and validation of other types of AdES digital signatures.

Procedures for creation, augmentation, and validation of CB-AdES digital signatures are out of scope.

The present multi-part deliverable aims at supporting electronic signatures independent of any specific regulatory framework.

NOTE 2: Specifically, but not exclusively, it is the aim that CB-AdES digital signatures specified in the present multi-part deliverable can be used to meet the requirements of electronic signatures, advanced electronic signatures, qualified electronic signatures, electronic seals, advanced electronic seals, and qualified electronic seals as defined in Regulation (EU) No 910/2014 [i.1].

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found in the ETSI docbox.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are necessary for the application of the present document.

[1] IETF RFC 8949 (December 2020): "Concise Binary Object Representtion (CBOR)".

[2] IETF RFC 9052 (August 2022): "CBOR Object Signing and Encryption (COSE): Structures and Process".

[3] IETF RFC 9360 (February 2023): "CBOR Object Signing and Encryption (COSE): Header Parameters for Carrying and Referencing X.509 Certificates".

[4] IETF RFC 9053 (August 2022): "CBOR Object Signing and Encryption (COSE): Initial Algorithms".

[5] IETF RFC 8610 (June 2019): "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Bingary Object Representation (CBOR) and JSON Data Structures".

[6] IETF RFC 9338 (December 2022): "CBOR Object Signing and Encryption (COSE): Countersignatures".

[7] IETF RFC 3061 (February 2001): "A URN Namespace of Object Identifiers".

[8] ETSI EN 319 132-1: "Electronic Signatures and Trust Infrastructures (ESI); XAdES digital signatures; Part 1: Building blocks and XAdES baseline signatures".

[9] ETSI TS 119 182-1: "Electronic Signatures and Trust Infrastructures (ESI); JAdES digital signatures; Part 1: Building blocks and JAdES baseline signatures".

[10] IETF RFC 5035 (August 2007): "Enhanced Security Services (ESS) Update: Adding CertID Algorithm Agility".

[11] Recommendation ITU-T X.509: "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks".

[12] IETF RFC 3161 (August 2001): "Internet X.509 Public Key Infrastructure Time Stamp Protocol (TSP)".

[13] IETF RFC 5280 (May 2008): "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[14] IETF RFC 6960 (June 2013): "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP".

[15] IETF RFC 5816 (April 2010): "ESSCertIDv2 Update for RFC 3161".

[16] IETF RFC 9597 (June 2024): "CBOR Web Token (CWT) Claims in COSE Headers".

[17] IETF RFC 4648 (October 2006): "The Base16, Base32, and Base64 Data Encodings".

[18] IETF RFC 8932 (May 2018): "CBOR Web Token (CWT)".

[19] ETSI TS 119 312 (V1.3.1): "Electronic Signatures and Infrastructures (ESI); Cryptographic Suites".

[20] IETF RFC 3986 (January 2005): "Uniform Resource Identifier (URI): Generic Syntax".

[21] IETF RFC 2616 (June 1999): "Hypertext Transfer Protocol - HTTP/1.1".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents may be useful in implementing an ETSI deliverable or add to the reader's understanding, but are not required for conformance to the present document.

[i.1] Regulation (EU) 2024/1183 of the European Parliament and of the Council of 11 April 2024 amending Regulation (EU) No 910/2014 as regards establishing the European Digital Identity Framework.

[i.2] ETSI EN 319 122-1: "Electronic Signatures and Infrastructures (ESI); CAdES digital signatures; Part 1: Building blocks and CAdES baseline signatures".

[i.3] ETSI EN 319 102-1: "Electronic Signatures and Trust Infrastructures (ESI); Procedures for Creation and Validation of AdES Digital Signatures; Part 1: Creation and Validation".

[i.4] ETSI TR 119 000: "Electronic Signatures and Infrastructures (ESI); The framework for standardization of digital signatures and trust services; Overview".

[i.5] ETSI TR 119 001: "Electronic Signatures and Infrastructures (ESI); The framework for standardization of signatures; Definitions and abbreviations".

[i.6] ETSI TR 119 100: "Electronic Signatures and Infrastructures (ESI); Guidance on the use of standards for signature creation and validation".

[i.7] ETSI TS 119 172-1: "Electronic Signatures and Infrastructures (ESI); Signature Policies; Part 1: Building blocks and table of contents for human readable signature policy documents".

[i.8] OASIS Standard: "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0".

[i.9] ETSI TS 101 533-1: "Electronic Signatures and Infrastructures (ESI); Data Preservation Systems Security; Part 1: Requirements for Implementation and Management".

[i.10] IETF RFC 4998: "Evidence Record Syntax (ERS)".

[i.11] W3C® Recommendation (19 November 2019): "Verifiable Credentials Data Model 1.0".

[i.12] draft-cavage-http-signatures-10 (May 2018): "Signing HTTP Messages".

[i.13] IETF RFC 7517 (May 2015): "CBOR Web Key (JWK)".

[i.14] ISO 3166-1: "Codes for the representation of names of countries and their subdivisions — Part 1: Country code".

3 Definition of terms, symbols, abbreviations and terminology

3.1 Terms

For the purposes of the present document, the terms given in ETSI TR 119 001 [i.5], IETF RFC 9052 [2] and the following apply:

byte string: sequence of octets (8 bits)

CB-AdES signature: COSE signature meeting the requirements specified in this or other parts of the present multi-part document

CBOR-bstr-wrapped: CBOR object wrapped in a CBOR byte string (bstr)

CBOR byte string: data item of CBOR major type 2

NOTE: The present document uses the term "bstr" for denoting a CBOR byte string.

COSE signature: CBOR structure containing a digitally signed message

NOTE: As specified in IETF RFC 9052 [2].

COSE Payload: binary data object (message(s), document(s)) that is signed by a COSE signature

NOTE 1: The COSE Payload bytes is only one of the components that contribute to generate the input byte string to the COSE signature value computation process, along with the protected headers sierialized map and the externally supplied data. See IETF RFC 9052 [2], clause 4.3 for externally supplied data, and IETF RFC 9052 [2], clause 4.4 for a specification of the signing process.

NOTE 2: The present document allows for a CB-AdES signature to simultaneously sign a set of more than one binary data objects. Therefore the COSE Payload may have more than one components. See clause 5.2.8 (sigD header parameter) of the present document.

NOTE 3: The present document allows for COSE Payload to be attached or detached. See clause 5.2.8 (sigD header parameter) of the present document.

COSE signature value: digital signature cryptographic value calculated over a sequence of octets derived from the protected headers serialized map, the COSE Payload, and the externally supplied data encapsulated in a CBOR byte string (bstr herein after)

NOTE 1: IETF RFC 9052 [2] does not provide any formally defined term for this object; instead it uses the term "signature value". The present document does not use this term, but "COSE signature value", for the sake of terminological coherence of other AdES specifications.

NOTE 2: The present document uses the term **CBOR Object Signing and Encryption (or its abbreviation, COSE)**, as defined by IETF RFC 9052 [2], i.e. for denoting the CBOR data structure for representing a digitally signed message.

NOTE 3: A CB-AdES signature is a special type of CBOR Object Signing and Encryption signature (or COSE signature). Therefore these terms are not directly interchangeable as, indeed a CB-AdES signature IS ALSO A COSE Signature, but NOT ALL COSE signatures are CB-AdES signatures.

electronic time-stamp: data in electronic form which binds other electronic data to a particular time establishing evidence that these data existed at that time

NOTE 1: In the case of IETF RFC 3161 [12] protocol, updated by IETF RFC 5816 [15], the electronic time-stamp is referring to the `timeStampToken` field within the `TimeStampResp` element (the TSA's response returned to the requesting client).

NOTE 2: This definition makes CB-AdES signatures not to be bound to a particular format of electronic time-stamp, because header parameters `adoTst`, `sigTst`, `arcTst`, `rfsTst`, and `sigRTst` can contain electronic time-stamps of any format.

payload field: payload field of the `COSE_Sign` CBOR array as specified in clause 4.1 of IETF RFC 9052 [2], or of the payload field of the `COSE_Sign1` CBOR array as specified in clause 4.2 of IETF RFC 9052 [2]

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ASCII	American Standard Code For Information Interchange
ASN.1	Abstract Syntax Notation 1
bstr	byte string
CA	Certification Authority
CBOR	Concise Binary Object Representation
CDDL	Concise Data Definition Language
COSE	CBOR Object Signing and Encryption
CRL	Certificate Revocation List
FIPS	Federal Information Processing Standards
HTTP	Hyper Text Transfer Protocol

IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union - Telecommunication
JSON	JavaScript Object Notation
OCSP	Online Certificate Status Protocol
OID	Object IDentifier
PKI	Public Key Infrastructure
RFC	Request For Comments
SAML	Security Assertion Markup Language
SHA	Secure Hash Algorithm
SIM	Subscriber Identification Module
SPO	Service Provision Option
tstr	text string
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UTC	Coordinated Universal Time

3.4 Terminology

The present document adopts, wherever it is possible the same terminology as the terminology used in IETF RFC 9052 [2].

Therefore, within the present document, the term "COSE Signature" shall denote the CBOR structure for digital signatures specified in IETF RFC 9052 [2].

The present document uses the term "CBOR data item" or just "data item" for denoting any of the data items specified in clause 2 of IETF RFC 8949 [1].

The present document uses the term "header parameter" for denoting a CBOR data item which is member either of the protected headers map or the unprotected headers map of COSE as specified in IETF RFC 9052 [2].

The present document uses the term "member" of a map for denoting one pair of CBOR data items (key, value) within the CBOR map (specified in IETF RFC 8949 [1].

The present document uses the term "element" or "element of the array" for denoting the contents of a position within a CBOR array (specified in of IETF RFC 8949 [1]).

NOTE: These last terms will be used for denoting each of the CBOR data items that will be added to the `uHeaders` CBOR array (specified in clause 5.3.1 of the present document), which will be incorporated in the unprotected headers set as a header parameter. Therefore, these CBOR data items will play, within the present document, an equivalent role to the role played by the unsigned attributes in CAdES and the unsigned qualifying properties in XAdES.

The present document uses the term "CB-AdES component" or "component" for denoting any CB-AdES signature constituent, regardless it is a header parameter, a member of a CBOR map, an element of a CBOR array, or any other CBOR data item.

IETF RFC 9052 [2] defines two structures for COSE signatures, namely: `COSE_Sign` (which allows several signatures on the same COSE Payload), and `COSE_Sign1` (which only allows one signature on the COSE Payload).

Below follows a copy of the CDDL specification for the `COSE_Sign` structure:

```
COSE_Sign = [
    Headers,
    payload : bstr / nil,
    signatures : [+COSE_Signature]
]
COSE_Signature = [
    Headers,
    signature : bstr
]
```

COSE_Sign structures can include header parameters in two layers:

1) Body layer. The header parameters appear within the first element (Headers) of the COSE_Sign CBOR array.

2) Signer layer. This layer is formed by the different elements in the signatures CBOR array, each one being a COSE_Signature array. The signer layer header parameters are included within the first element (Headers) of the COSE_Signature array.

Below follows a copy of the CDDL specification for the COSE_Sign1 structure:

```
COSE_Sign1 = [
    Headers,
    payload : bstr / nil,
    signature :bstr
]
```

COSE_Sign1 structures have only one layer: the body layer. Therefore, COSE_Sign1 structures include all the header parameters only in the body layer.

The present document uses this special font for denoting the names of CB-AdES components and COSE structures supporting CB-AdES signatures.

4 General Requirements

4.1 CDDL definitions

The present document defines the new types and components for CB-AdES signatures using the Concise Data Definition Language (CDDL) [5].

4.2 Requirements on CB-AdES supporting COSE structures

CB-AdES signatures specified in the present document may be built on both types of COSE signature structures specified in IETF RFC 9052 [2], namely: COSE_Sign for multiple signers of the same COSE Payload (specified in clause 4.1 of IETF RFC 9052 [2]), and COSE_Sign1 for one single signer (specified in clause 4.2 of IETF RFC 9052 [2]).

4.3 Requirements on CB-AdES encoding

CB-AdES signatures specified in the present document may be encoded as untagged (COSE_Sign and COSE_Sign1) or tagged, namely COSE_Sign_Tagged (specified in clause 4.1 of IETF RFC 9052 [2]), and COSE_Sign1_Tagged (specified in clause 4.2 of IETF RFC 9052 [2]).

4.4 Requirements on CB-AdES headers

The unprotected headers map in CB-AdES signatures (regardless they are within the body layer or the signer layer) shall contain only one member, namely the uHeaders header parameter (specified in clause 5.3 of the present document), which is defined as a CBOR array.

NOTE 1: The rationale for this is that the unprotected header is a CBOR map, and no order may be inferred in its different members. This is the reason why the present document defines uHeaders header parameter as a CBOR array.

NOTE 2: The elements of this CBOR array will contain CBOR values that play for CB-AdES signatures the same role as the role played by the unsigned header parameters for JAdES signatures.

NOTE 3: An immediate consequence is that a time-stamp token present within the `arcTst` component specified in clause 5.3.5 of the present document, protects the COSE Payload, the protected headers map, the COSE signature value, and the `uHeaders` header parameter within the unprotected headers map.

CB-AdES signatures supported by a `COSE_Sign` structure:

1) May include protected header parameters in both, the body layer and the signer layer.

2) Shall not contain the `uHeaders` unprotected header parameter in the body layer.

3) May include the `uHeaders` unprotected header parameter in the signer layer.

CB-AdES signatures supported by a `COSE_Sign1` structure shall include header parameters at the body layer (as they do not have signer layer).

New header parameters defined in the present document, or already defined elsewhere but further profiled in the present document, shall be incorporated into the CB-AdES signature as specified in the present document.

NOTE 4: For each header parameter newly defined or defined elsewhere but further specified by the present document, the present document specifies if it is signed or/and unsigned, and, in the case of CB-AdES signatures supported by `COSE_Sign` structures, the layer where it is placed.

Header parameters defined elsewhere and not further profiled by the present document, may also be added as signed header parameters or as elements of `uHeaders` header parameter within the CB-AdES, in accordance with the requirements defined in the present document (for example, those applicable to CB-AdES signatures supported by a `COSE_Sign` structure). Their semantics and processing are out of the scope of the present document.

4.5 Requirements on Payload

In CB-AdES signatures, the COSE Payload may be attached or detached.

Detached COSE Payload may either be one detached object, or result from the concatenation of more than one detached data objects. See the specification of `sigD` signed header parameter in clause 5.2.8 of the present document.

4.6 Requirements on map keys and data tags

The present document specifies new CBOR types with the following criteria:

- The keys of the CBOR maps pairs shall be integers.
- Tags for new CBOR tagged data items shall also be integers.

4.7 Requirements on encapsulation in CBOR byte strings

The present document requires to encapsulate CBOR components in CBOR byte strings.

This encapsulation shall be performed using the CBOR encoding restrictions defined in clause 9 of IETF RFC 9052 [2].

5 Header parameters semantics and syntax

5.1 Header parameters defined by IETF

5.1.1 Introduction

This clause defines additional requirements for the use of some of header parameters specified by IETF.

NOTE: Clause 6.3 specifies requirements (mainly of presence and cardinality), for the use of some of the header parameters specified by IETF for CB-AdES baseline signatures.

5.1.2 The `alg` (algorithm) header parameter

Semantics

The `alg` header parameter shall be a signed header parameter that qualifies the signature.

The `alg` header parameter shall have the semantics specified in IETF RFC 9052 [2], clause 3.1.

Syntax

The `alg` header parameter shall have the syntax specified in IETF RFC 9052 [2], clause 3.1.

Its value should be one of the algorithms for digital signatures recommended by in ETSI TS 119 312 [19].

The identifier of the algorithm shall be one of the identifiers registered at the IANA "COSE Algorithms" (https://www.iana.org/assignments/cose/cose.xhtml#algorithms).

In CB-AdES signatures supported by a `COSE_Sign` structure, this header parameter shall be placed at the signer layer.

5.1.3 The `content type` (content type) header parameter

Semantics

The `content type` header parameter shall be a signed header parameter that qualifies the COSE Payload.

The `content type` header parameter shall have the semantics specified in IETF RFC 9052 [2], clause 3.1.

The `content type` header parameter shall not be present if the `sigD` header parameter, specified in clause 5.2.8 of the present document, is present within the CB-AdES signature.

The `content type` header parameter should not be present if the content type is implied by the COSE Payload.

The `content type` header parameter shall not be present if the COSE Payload is a (counter-signed) signature.

Syntax

The `content type` header parameter shall have the syntax specified in IETF RFC 9052 [2], clause 3.1.

In CB-AdES signatures supported by a `COSE_Sign` structure, this header parameter shall be placed at the signer layer.

NOTE: This is because this header parameter qualifies the COSE Payload.

5.1.4 The `kid` (key identifier) header parameter

Semantics

The `kid` header parameter shall be a signed header parameter that qualifies the signature.

The `kid` header parameter shall have the semantics specified in IETF RFC 9052 [2], clause 3.1.

The content of `kid` header parameter should be the DER-encoded instance of type `IssuerSerial` type defined in IETF RFC 5035 [10].

The header parameter `kid` shall be used as a hint that can help to identify the signing certificate if other header parameters referencing or containing the signing certificate are present in the CB-AdES signature.

Syntax

The `kid` header parameter shall have the syntax specified in IETF RFC 9052 [2], clause 3.1.

In CB-AdES signatures supported by a `COSE_Sign` structure, this header parameter shall be placed at the signer layer.

5.1.5 The x5u (X.509 URL) header parameter

Semantics

The x5u header parameter shall be a signed header parameter that qualifies the signature.

The x5u header parameter shall have the semantics specified in IETF RFC 9360 [3], clause 2.

The x5u member shall be used as a hint, as implementations can have alternative ways for retrieving the referenced certificate if it is not found at the referenced place.

Syntax

The x5u header parameter shall have the syntax specified in IETF RFC 9360 [3], clause 2.

In CB-AdES signatures supported by a COSE_Sign structure, this header parameter shall be placed at the signer layer.

5.1.6 The CBOR component for counter signatures

Semantics

The CBOR component for including one or more counter signatures shall be an element of the uHeaders member of the unprotected headers map (see clause 5.3.1 of the present member for the specification of uHeaders member).

The element of the uHeaders member of the unprotected headers map used for including one or more counter signatures shall be either one of the two header parameters specified in IETF RFC 9338 [6], or a CB-AdES signature as specified in the present document.

Syntax

Each CBOR component for incorporating new counter signatures shall contain either one of the two header parameters specified in IETF RFC 9338 [6].

NOTE 1: The use of a counter signature as specified in IETF RFC 9338 [6] or a CB-AdES as a counter signature is use-case or policy dependent. Therefore, the present document does not make any recommendation in this sense. Discriminating whether the counter signature is a CB-AdES can be done by checking its sets of protected and unprotected header parameters; if the contents of these sets are according to the requirements in Table 14 in clause 6.3 of the present document, then the counter signature is a CB-AdES signature. Otherwise, it is a non CB-AdES counter signature.

The digital signature value of each counter signature shall be computed as specified in clause 3.3 of IETF RFC 9338 [6].

If the counter signature is a CB-AdES signature, its digital signature value shall be computed as specified in clause 3.3 of IETF RFC 9338 [6] In these cases the context string shall be selected according to the structure used by the CB-AdES counter signature.

NOTE 2: Clause 4.4 states that CB-AdES signatures supported by a COSE_Sign structure, only have unprotected header in the signer layer.

5.1.7 The x5t (cert hash) header parameter

Semantics

The x5t header parameter shall have the semantics specified in IETF RFC 9360 [3], clause 2.

Syntax

The x5t header parameter shall have the syntax of COSE_CertHash type, specified in IETF RFC 9360 [3], clause 2.

```
COSE_CertHash = [ hashAlg: (int / tstr), hashValue: bstr ]
```

NOTE: The `x5t` component is an array of two elements. The first one identifies a digest algorithm. The second one contains the digest value of a certificate DER-encoded. See in IETF RFC 9360 [3], clause 2 for further details on the types of each component of this array.

The value of the hash identifier (first element in the CBOR array -`hashAlg`) shall be one of the identifiers for digest algorithms registered in IANA COSE Algorithms registry (https://www.iana.org/assignments/cose/cose.xhtml#algorithms), or any future specification that defines new identifiers for digest algorithms.

In CB-AdES signatures supported by a `COSE_Sign` structure, this header parameter shall be placed at the signer layer.

5.1.8 The `x5chain` (X.509 chain) header parameter

Semantics

The `x5chain` header parameter may be a signed or unsigned header parameter that qualifies the signature.

If `x5chain` has not to be signed, it shall be included within the `uHeaders` CBOR array specified in clause 5.3.1 of the present document.

NOTE: This is for meeting requirements appearing in use cases that require to sign the digest of the signing certificate but not the signing certificate itself, while also requiring to include it within the CB-AdES signature.

The `x5chain` header parameter shall have the semantics specified in IETF RFC 9360 [3], clause 2.

Syntax

The `x5chain` header parameter shall have the syntax of `COSE_X509` type specified in IETF RFC 9360 [3], clause 2.

```
COSE_X509 = bstr / [ 2*certs: bstr ]
```

If the `x5chain` is unsigned, it shall be placed within the `uHeaders` CBOR array (and therefore within the unprotected header).

In CB-AdES signatures supported by a `COSE_Sign` structure, this header parameter shall be placed in the signer layer.

5.1.9 The `iat` (issued at) header parameter

Semantics

The `iat` header parameter may be a signed or unsigned header parameter that qualifies the signature.

The `iat` header parameter shall have the semantics specified in clause 3.1.6 of IETF RFC 8392 [18].

The `iat` header parameter's value shall specify the time at which the signer claims to have performed the signing process.

Syntax

The syntax of the `iat` header parameter shall be as specified in IETF RFC 8932 [18], clause 3.1.6.

Its value shall be an instance of NumericDate, which is specified in IETF RFC 8932 [18], clause 2.

The `iat` header parameter shall be incorporated to CB-AdES signature within the CWT claim specified in IETF RFC 9597 [16]. This CWT claim shall be incorporated to the CB-AdES signature as a signed header parameter as specified in IETF RFC 9597 [16].

In CB-AdES signatures supported by a `COSE_Sign` structure, this header parameter shall be placed at the signer layer.

5.1.10 The `crit` (critical) header parameter

Semantics

The `crit` header parameter shall be a signed header parameter that qualifies the signature.

The `crit` header parameter shall have the semantics specified in IETF RFC 9052 [2], clause 3.1.

Syntax

The `crit` header parameter shall have the syntax specified in IETF RFC 9052 [2], clause 3.1.

If the CB-AdES signature includes the `sigD` header parameter, the `crit` header parameter shall also be present and the label assigned to the `sigD` header parameter shall be one of its CBOR array elements.

In CB-AdES signatures supported by a `COSE_Sign` structure, this header parameter shall be placed at the signer layer.

5.2 New signed header parameters

5.2.1 Labels and tags of the signed header parameters

Clause 5.2 specifies a number of signed header parameters. All of them shall be part of the protected headers map when present. All of them shall be identified by a label in the corresponding CBOR map that shall be an integer.

Table 1 below defines the labels associated to each signed header parameter and the tags identifying each signature policy qualifier specified in clause 5.2.

Table 1: Tags for signed header parameters specified in the present document

Signed header parameter type name	Signed header parameter label and qualifiers tags
`x5ts`	`261`
`srCms`	`262`
`sigPl`	`263`
`srAts`	`264`
`adoTst`	`265`
`sigPId`	`266`
`sigD`	`267`

The following CDDL rules assign the integer values of the tags of the signed header parameters to identifiers that will be used throughout the rest of the present document for making the CDDL rules easier to read.

NOTE: In the CDDL rules that appear below the `_l` stands for "label".

```
x5ts_l = 261
srCms_l = 262
sigPl_l = 263
srAts_l = 264
adoTst_l = 265
sigPId_l = 266
sigD_l = 267

ETSI_Signed_Headers = (
    ? 33 => x5chain,    ;X.509 cert path or signing certificate
    ? x5ts_l => x5ts,   ;Reference to signing certificate / certs in cert path
    ? srCms_l => srCms, ;Signer commitments
    ? sigPl_l => sigPl, ;Signature production place
    ? srAts_l => srAts, ;Signer attributes
    ? adoTst_l => adoTst,   ;COSE payload time-stamp
    ? sigPId_l => sigPId,   ;Signature Policy Identifier
    ? sigD_l => sigD    ;Detached COSE Payload reference data
)
```

5.2.2 The `x5ts` (X.509 certificates Thumbprints) header parameter

Semantics

The `x5ts` header parameter shall be a signed header parameter that qualifies the signature.

Its label shall be `261`.

The x5ts header parameter shall contain several references of certificates within the certification path of the signing certificate, each one formed by the identifier of a digest algorithm and the digest value of the referenced certificate.

The first reference within the x5ts header parameter shall be the reference of the signing certificate.

The x5ts header parameter shall not contain any other information.

Syntax

Below follows the CDDL definition of the x5ts header parameter:

```
x5ts = [2*x5t: COSE_CertHash]
```

Each x5t component of x5ts shall have the semantics and syntax specified in IETF RFC 9360 [3], clause 2 and further profiled in clause 5.1.7 of the present document.

A CB-AdES signature shall have at least one of the following header parameters in the protected headers map: x5t, x5ts, or x5chain (specified in IETF RFC 9360 [3], clause 2 and further profiled in clause 5.1.8 of the present document) for protecting the signing certificate with the signature.

In CB-AdES signatures supported by a COSE_Sign structure, this header parameter shall be placed at the signer layer.

5.2.3 The srCms (signer commitments) header parameter

Semantics

The srCms header parameter shall be a signed header parameter that qualifies the COSE Payload.

Its label shall be 262.

The srCms header parameter shall indicate the commitment made by the signer when signing.

The srCms header parameter shall express the commitment type with a URI.

The srCms header parameter may contain a sequence of qualifiers providing more information about the commitment.

A commitment type associated to a signature is, as specified in clause A.3.2.2 of ETSI TS 119 172-1 [i.7], "*the representation of the expected purpose and meaning of the signature and of the precise nature of the responsibility assumed by the signer when generating the concerned signature*".

NOTE 1: The commitment type can be:

- defined as part of the signature policy, in which case, the commitment type has precise semantics that are defined as part of the signature policy; or
- be a registered type, in which case, the commitment type has precise semantics defined by registration, under the rules of the registration authority. Such a registration authority can be a trading association or a legislative authority.

NOTE 2: Annex B of ETSI TS 119 172-1 [i.7] has defined a set of commitment types and their corresponding URIs. These URIs are listed in Annex C of the present document.

Syntax

Below follows the CDDL definition of the srCms header parameter:

```
srCms = [+SrCm]
SrCm = {
    1 => oId,   ;commId the commitment identifier: an oId data type
    ? 2 => [+any]   ;commQuals: qualifiers
}
```

Each element of the srCms CBOR array shall indicate one commitment made by the signer, which may be further qualified.

The `commId` member of the `SrCm` CBOR map is an instance of `oId` type (a CBOR map), which is specified in clause 5.4.1 of the present document. The `id` member of `oId` shall have a URI as value, uniquely identifying one commitment made by the signer.

The `commQuals` member of the `SrCm` CBOR map provides means to include additional qualifying information on the commitment made by the signer.

NOTE 3: None of the commitment types defined in Annex B of ETSI TS 119 172-1 [i.7] have commitment qualifiers.

The specification of additional qualifying information of the commitment is out of the scope of the present document. Any specification defining a new commitment type that requires additional qualifying information, shall provide a full definition of the semantics and the syntax of the mentioned qualifying information.

Table 2 shows the values assigned to each of the keys in the maps specified in the present clause.

Table 2: Values of keys in maps specified in the present clause

Name	Key value in map
`commId`	1
`commQuals`	2

In CB-AdES signatures supported by a `COSE_Sign` structure, this header parameter shall be placed at the signer layer.

5.2.4 The `sigPl` (signature production place) header parameter

Semantics

The `sigPl` header parameter shall be a signed header parameter that qualifies the signer.

Its label shall be `263`.

The `sigPl` header parameter shall specify an address associated with the signer at a particular geographical (e.g. city) location.

Syntax

Below follows the CDDL definition of the `sigPl` header parameter:

NOTE: Its definition follows the specification of `PostaAddress` type in schema.org (https://schema.org/PostalAddress).

```
sigPl = {
        ? 1 => tstr,    ; addressCountry
        ? 2 => tstr,    ; addressLocality
        ? 3 => tstr,    ; addressRegion
        ? 4 => tstr,    ; postOfficeBoxNumber
        ? 5 => tstr,    ; postalCode
        ? 6 => tstr,    ; streetAddress
}
```

The `sigPl` header parameter shall have at least one of its members.

The `addressCountry` member shall contain may contain either the name of the country or its two-letter ISO 3166-1 [i.14] alpha-2 country code.

Table 3 shows the values assigned to each of the keys in the maps specified in the present clause.

Table 3: Values of keys in maps specified in the present clause

Name	Label in map
addressCountry	1
addressLocality	2
addressRegion	3
postOfficeBoxNumber	4
postalCode	5
streetAddress	6

In CB-AdES signatures supported by a COSE_Sign structure, this header parameter shall be placed at the signer layer.

5.2.5 The srAts (signer attributes) header parameter

Semantics

The srAts header parameter shall be a signed header parameter that qualifies the signer.

Its label shall be 264.

The srAts header parameter shall encapsulate signer attributes (e.g. role). This header parameter may encapsulate the following types of attributes:

- attributes claimed by the signer;
- attributes certified in attribute certificates issued by an Attribute Authority; or/and
- assertions signed by a third party.

Syntax

Below follows the CDDL definition of the srAts header parameter:

```
srAts = {
    ? 1 => CertifiedAttrs,  ;certified: Certified signer attributes
    ? 2 => AttrArrays,  ; signedAssertions: Signed assertions for signer
    ? 3 => AttrArrays   ; claimed: Claimed signer attributes
}

CertifiedAttrs = [ + CertifiedAttr]

CertifiedAttr = {CertifiedAttrChoice}

CertifiedAttrChoice = (
    1 => pkiObj //  ;x509AttrCert: encapsulates a X.509 attribute certificate
    2 => pkiObj     ; otherAttrCert: encapsulates another type of attribute certificate
)

AttrArrays = [+NotCertifiedItem]

label = int / tstr
value = any

NotCertifiedItem = [
    mediaType : tstr,   ;String identifying the media type of claimed attributes or signed
                       ;assertions
    *label => any   the not certified item
]
```

The certified member shall contain a non-empty array of certified attributes, which shall be one of the following:

- the DER-encoded X.509 attribute certificates conformant to Recommendation ITU-T X.509 [11] issued to the signer, within the x509AttrCert member; or
- attribute certificates (issued, in consequence, by Attribute Authorities) in different syntax than the one specified in Recommendation ITU-T X.509 [11], within the otherAttrCert member. The definition of specific otherAttrCert is outside of the scope of the present document.

The `signedAssertions` member shall contain a non-empty array of assertions signed by a third party.

NOTE 1: A signed assertion is stronger than a claimed attribute, since a third party asserts with a signature that the attribute of the signer is valid. However, it is less restrictive than an attribute certificate.

EXAMPLE: Verifiable credentials as specified in W3C® Recommendation [i.11] if serialized in CBOR.

The `claimed` member shall contain a non-empty array of attributes claimed by the signer but neither certified by an Attribute Authority, nor signed by any entity issuing assertions.

Both the `signedAssertions` and the `claimed` members shall be instances of `AttrArrays` type. Each instance of this type shall be a CBOR array whose elements are instances of `NotCertifiedItem` CBOR array. Each instance of `NotCertifiedItem` shall contain two elements, whose contents are specified below:

a) The `mediaType` shall contain a string identifying the media type of the signed assertions or the claimed attributes present in `qVals` member.

NOTE 2: The media types registered by IANA can be found at https://www.iana.org/assignments/media-types/media-types.xhtml#app4lication. Nowadays, the media type registered also include information on the character-set encoding.

b) The `qVals` member, which shall be a CBOR array of at least one item. The elements of `qVals` CBOR array shall be the values of the signed assertions or the claimed attributes encoded as indicated within the `encoding` member.

NOTE 3: Instances of `AttrArrays` type allow incorporating signed assertions and/or claimed attributes of different types.

The definition of specific content types for `signedAssertions` and `claimed` members is outside of the scope of the present document.

Empty `srAts` header parameters shall not be generated.

Table 4 shows the values assigned to each of the keys in the maps or groups specified in the present clause.

Table 4: Values of keys in maps or groups specified in the present clause

Name	Label in map
`certified`	1
`signedAssertions`	2
`claimed`	3
`x509AttrCert (in CertifiedAttrChoice)`	1
`otherAttrCert (in CertifiedAttrChoice)`	2

In CB-AdES signatures supported by a `COSE_Sign` structure, this header parameter shall be placed at the signer layer.

5.2.6 The `adoTst` (COSE payload time-stamp) header parameter

Semantics

The `adoTst` header parameter shall be a signed header parameter that qualifies the COSE Payload.

Its label shall be `265`.

The `adoTst` header parameter shall encapsulate one or more electronic time-stamps, generated before the signature production, whose message imprint computation input shall be the COSE Payload of the CB-AdES signature.

Syntax

Below follows the CDDL definition of the `adoTst` header parameter:

```
adoTst = tstContainer
```

The message imprint computation input for the time-stamp token shall be an octet stream built as indicated below:

1) If the `sigD` header parameter, as specified in clause 5.2.8 of the present document, is absent then the message imprint computation input shall be:

   - The CBOR byte string of the `payload` field, if the `payload` field is present.

   - The bytes of the detached COSE Payload, encapsulated in a CBOR byte string, if the COSE Payload is detached (the `payload` field is absent).

2) Else, if the `sigD` header parameter is present, if the value of its `mId` member is "`http://uri.etsi.org/19152/ObjectIdByURI`" or "`http://uri.etsi.org/19152/ObjectIdByURIHash`" then concatenate the bytes resulting from processing the contents of its `pars` member as specified in clause 5.2.8.2.2 of the present document.

NOTE: The rationale for applying the processing specified in clause 5.2.8.2.2 of the present document to the case of the mechanism identified by "`http://uri.etsi.org/19152/ObjectIdByURIHash`" is the fact that this is an indirect signing mechanism, i.e. based on signing digest values of data objects, instead of the data objects themselves. Time-stamping not the digest values but the retrieved data objects, protects against future weaknesses of the digest algorithms used in `sigD`.

3) Else, if the value of its `mId` member is neither "`http://uri.etsi.org/19152/ObjectIdByURI`" nor "`http://uri.etsi.org/19152/ObjectIdByURIHash`", then it is out of the scope of the present document to specify how to retrieve the COSE Payload, and the specification defining the value of the `mId` member shall have to specify how to retrieve the COSE Payload.

If the COSE Payload is detached and the CB-AdES signature does not incorporate the `sigD` signed header parameter, then it is out of the scope to specify how to retrieve the COSE Payload.

In CB-AdES signatures supported by a `COSE_Sign` structure, this header parameter shall be placed at the signer layer.

5.2.7 The `sigPId` (signature policy identifier) header parameter

5.2.7.1 Semantics and syntax

Semantics

The `sigPId` header parameter shall be a signed header parameter qualifying the signature.

Its label shall be `266`.

The `sigPId` header parameter shall contain an explicit identifier of a signature policy.

NOTE: ETSI TS 119 172-1 [i.7] specifies a framework for signature policies.

Syntax

Below follows the CDDL definition of the `sigPId` header parameter:

```
sigPId = {
    1 => oId,                   ;id: instance of oId type identifying the signature policy
    2 => DigAlgVal,             ;digAlgVal: digest algorithm and value of the signature policy
                                ;document
    ? 3 => bool .default false, ;digPSp: indicates whether the digest has been computed according to
                                ;some spec, default value: false
    ? 4 => [+SigPQual]          ;sigPQuals: signature policy qualifiers
}

DigAlgVal = [ hashAlg: (int / tstr), hashValue: bstr ]
```

The `id` member shall be used for referencing the signature policy explicitly. It shall uniquely identify a specific version of the signature policy.

The `digAlgVal` component shall be a CBOR array. Its first element shall be one of the identifiers for digest algorithms registered in IANA COSE Algorithms registry (https://www.iana.org/assignments/cose/cose.xml#algorithms), https://www.iana.org/assignments/cose/cose.xml#header-algorithm-parameters or one of the identifiers for digest algorithms defined in IETF RFC 9053 [4], or any future specification that amends, complements, or supersedes it. Its second element shall be the value of the digest computed on the signature policy document using the algorithm identified in the first element of the CBOR array.

The `digPSp` member shall be a CBOR Boolean value. When present and set to `"true"`, it shall indicate that the digest of the signature policy document has been computed as specified in a technical specification. Absence of this member shall be considered as if present and set to `"false"`. If this member is present and set to `"true"`, then the `spDSpec` qualifier shall be present and shall identify the aforementioned technical specification.

The `sigPQuals` member shall be a non-empty array of qualifiers of the signature policy.

Clause 5.2.7.2 specifies three signature policy qualifiers.

Table 5 shows the values assigned to each of the keys in the maps specified in the present clause.

Table 5: Values of keys in maps or groups specified in the present clause

Name	Label in map
`id`	1
`digAlgVal`	2
`digPSP`	3
`sigPQuals (in CertifiedAttrChoice)`	4

In CB-AdES signatures supported by a `COSE_Sign` structure, this header parameter shall be placed at the signer layer.

5.2.7.2 Signature policy qualifiers

Semantics

This clause specifies three qualifiers for the signature policy. Each qualifier shall be a CBOR tagged data item. The present document defines the following qualifiers and tags:

- A URL where a copy of the signature policy document can be obtained (`spURI` choice). Its tag shall be 1.
- A user notice that should be displayed when the signature is validated (`spUserNotice` choice). Its tag shall be 2.
- An identifier of the technical specification that defines the syntax used for producing the signature policy document (`spDSpec` choice). Its tag shall be 3.

Syntax

Below follows the CDDL definitions of the `spURI`, `spUserNotice`, and `spDSpec` qualifiers:

```
SigPQual = {
    ? 1 => #6.32(tstr) //              ;spURI: URL where a copy of the signature policy document
                                       ;can be obtained
    ? 2 => SpUserNotice//       ;spUserNotice: Info displayed when signature is validated
    ? 3 => SpDesc //            ;spDSpec: identifier of the technical specification that defines
                                ;the syntax used for producing the signature policy document
    *label => value         ;otherQuals: extension point for qualifiers not specified in
                                ;the present document. Reminder: label is defined in clause 5.2.5
                            ; either as an uint or a tstr, and value is defined as any.
}

SpUserNotice = {
    ? 1 => NoticeRef,   ; noticeRef: User notice and references
    ? 2 => tstr         ;explText: notice text to be displayed
}

NoticeRef = {
    1 => tstr,  ;org: the name of the organization
    2 => [+uint]    ;noticeNumbers: the notice numbers identifying textual statements
```

```
}

SpDSpec = obId
```

The `spURI` qualifier shall contain a URL value where a copy of the signature policy document can be obtained.

NOTE 1: This URL can reference, for instance, a remote site (which can be managed by an entity entitled for this purpose) from where (signing/validating) applications can retrieve the signature policy document.

The `spUserNotice` qualifier shall contain information that is intended for being displayed whenever the signature is validated. The `org` member shall indicate the name of the organization. The `noticeNumbers` shall be a CBOR array with unsigned integers.

At least one of the two members of `spUserNotice` qualifier shall be present.

The `explText` member shall contain the text of the notice to be displayed.

The `noticeRef` member shall name an organization and shall identify by numbers (`noticeNumbers` member) a group of textual statements prepared by that organization, so that the application could get the explicit notices from a notices file.

NOTE 2: Other notices can come from the organization issuing the signature policy.

The `spDSpec` member shall identify the technical specification that defines the syntax used for producing the signature policy document.

The `otherQuals` member shall be a non-empty CBOR array. Each element in the array shall be a qualifier.

NOTE 3: The `otherQuals` member is an extension point for adding qualifiers not specified in the present document.

Table 6 shows the values assigned to each of the keys in the maps specified in the present clause.

Table 6: Values of keys in maps or groups specified in the present clause

Name	Label in map
spUri	1
spUserNotice	2
spDSpec	3
otherQuals	4
noticeRef (in SPUserNotice)	1
explText (in SPUserNotice)	2
org (in NoticeRef)	1
noticeNumbers (in NoticeRef)	2

5.2.8 The `sigD` header parameter

5.2.8.1 Semantics and Syntax

Semantics

The `sigD` header parameter shall be a signed header parameter.

Its label shall be `267`.

The `sigD` header parameter shall not appear in CB-AdES signatures whose COSE Payload is attached.

The `sigD` header parameter may appear in CB-AdES signatures whose COSE Payload is detached.

A CB-AdES signature shall have at most one `sigD` header parameter within each present protected header .

NOTE 1: When using `COSE_Sign`, it is possible to build a COSE signature which allows multiple signers to sign the same COSE Payload. Each element of its `signatures` element includes its own protected header; therefore, each signer can include its own `sigD` header parameter.

The `sigD` header parameter shall:

1) Reference one or more detached data objects.

2) Specify how the aforementioned references shall be processed for contributing to build the sequence of octets that shall be the COSE Payload of the CB-AdES signature.

3) Allow defining different mechanisms for meeting the two aforementioned requirements.

4) Not be present as a header parameter of a counter signature, as it is clear what an embedded counter signature signs.

Chaining of references shall not be allowed. Only the data objects directly referenced within the `sigD` header parameter shall contribute to build the COSE Payload. If some referenced object contains in its turn references to other data objects, these last data objects shall not contribute to build the COSE Payload.

NOTE 2: This is for avoiding building trees of referenced and distributed data objects, which would complicate the validation of CB-AdES signatures.

The `sigD` header parameter may also incorporate digest values of the referenced data objects encapsulated within a CBOR byte string.

The `sigD` header parameter may also incorporate any additional information for meeting requirements 1) and 2) as required by the mechanisms mentioned in 3).

Syntax

Below follows the CDDL definition of the `sigD` header parameter:

```
sigD : {
    1 => #6.32(tstr),   ;mId: URI identifying the mechanism used for referencing and processing each
                        ;referenced data object
    2 => [+tstr],   ;pars: References to data objects as per the mechanism identified by mId
    ? 3 => (int / tstr) ;hashM: Digest algorithm identifier
    ? 4 => [+bstr], ;hashV: Digest values of referenced data objects as per algorithm identified by
                    ;hashM
    ? 5 => [+tstr]  ;ctys: Indication of the content type of each referenced object
}
```

The `mId` member shall be present. It shall be an URI identifying the mechanism used for referencing and processing each referenced data object for building the COSE Payload. The present document defines 2 referencing mechanisms with their corresponding identifiers in clauses 5.2.8.2.2 and 5.2.8.2.3.

The `pars` member shall be present. It shall be a non-empty array of strings. Each element of the array shall contain a reference to one data object, as required by the identification mechanism identified in the `mId` member.

The `hashM` member shall be the identifier of a digest algorithm. Its value shall be one of the identifiers for digest algorithms registered in IANA COSE Algorithms registry (https://www.iana.org/assignments/cose/cose.xml#algorithms), https://www.iana.org/assignments/cose/cose.xml#header-algorithm-parameters or one of the identifiers for digest algorithms defined in IETF RFC 9053 [4], or any future specification that amends, complements, or supersedes it.

The presence of the `hashM` member shall be conditional on the definition of the identification mechanism. If this member is present, then `hashV` member shall be present.

The `hashV` member shall be a non-empty array of strings. Each element of the array shall contain the digest value of the data object referenced by the parameter value that is present in the same position of the `pars` array, encapsulated within a CBOR byte string.

The presence of the `hashV` member shall be conditional on the definition of the identification mechanism. If this member is present, then `hashM` member shall be present.

The ctys member shall be a non-empty an array of strings. The contents of each element of this array shall have the same semantics of the content type header parameter specified in IETF RFC 9052 [2], clause 3.1.

There shall be as many elements within the ctys array as elements within the array pars. Each element of the ctys array shall contain the information corresponding to the data object referenced by the parameter value that is present in the same position of the pars array, except if the content type is implied by the data object or the data object is a counter-signed signature: in these cases, the element of the ctys array shall have as value a null (#7.22 or 0xF6).

Table 7 shows the values assigned to each of the keys in the maps specified in the present clause.

Table 7: Values of keys in maps or groups specified in the present clause

Name	Label in map
mId	1
pars	2
hashM	3
hashV	4
ctys	5

In CB-AdES signatures supported by a COSE_Sign structure, this header parameter shall be placed at the signer layer.

5.2.8.2 Mechanisms supported by URI-references

5.2.8.2.1 General requirements

This clause specifies two mechanisms that use URI-references for referencing the data objects contributing to build the COSE Payload.

For these referencing mechanisms, the contents of the pars member shall be an array of strings. Each string shall be an URI-reference, which, once resolved, shall result in an URI appertaining to the group of URIs that can be classified as locators according to clause 1.1.3 of IETF RFC 3986 [20]. Each URI-reference shall refer one data object.

NOTE: According to IETF RFC 3986 [20], URI references that can be classified as locators (URLs are the obvious example) "*provide a means of locating the resource by describing its primary access mechanism*".

When resolving an URI-reference which is a relative reference, conforming application shall set a default base HTTP scheme URI when applying clause 5.1.4 of IETF RFC 3986 [20].

Dereferencing URI-references in the HTTP scheme shall be supported. Dereferencing an URI-reference in the HTTP scheme shall comply with the Status Code Definitions specified in clause 10 of IETF RFC 2616 [21].

Dereferencing URI-references in other locator schemes may be supported. Dereferencing URI-references within one of such schemes shall be conducted as defined in the corresponding scheme specification.

5.2.8.2.2 Mechanism ObjectIdByURI

The URL identifying this referencing mechanism shall be "http://uri.etsi.org/19152/ObjectIdByURI".

For this referencing mechanism, neither hashV, nor hashM shall be present. Member ctys may be present.

The semantics and syntax of each element of array ctys shall be as specified in clause 5.2.8.1 of the present document.

The stream of octets corresponding to the contribution of the COSE Payload to the computation of the COSE signature value shall be generated as indicated below:

1) Initialize the stream of octets to an empty stream.

2) While there are URI-references in the pars array not visited:

   - Take the next one.

   - Dereference the URI referemce, as specified in clause 5.2.8.2.1 of the present document.

- Concatenate the resulting octets to the stream of octets that will form the COSE Payload.

5.2.8.2.3 Mechanism ObjectIdByURIHash

The URL identifying this referencing mechanism shall be "http://uri.etsi.org/19152/ObjectIdByURIHash".

For this referencing mechanism, the hashV, and the hashM members shall be present. Member ctys may be present.

The semantics and syntax of hashM, hashV, and ctys shall be as specified in clause 5.2.8.1 of the present document.

For computing the digest values that appear within the hashV member, each data object referenced within the pars member, shall be retrieved as specified in clause 5.2.8.2.1 of the present document.

When using this mechanism, the COSE Payload shall contribute as an empty stream to the computation of the COSE signature value.

NOTE 1: As this sigD is a signed header parameter, and it already includes the digest of the components of the COSE Payload, the COSE Payload is indirectly signed by signing the sigD signed header parameter, and consequently, this referencing mechanism does not require that the COSE Payload directly contributes to the computation of the COSE signature value.

If the COSE Payload is required for other purposes than computing the COSE signature value when this mechanism is used, it shall be generated as specified in clause 5.2.8.2.2.

NOTE 2: The generation of this COSE Payload is required, for instance, for generating the adoTst or the arcTst header parameters.

5.3 New unsigned header parameter

5.3.1 The uHeaders header parameter

5.3.1.1 Semantics and syntax

Semantics

The uHeaders parameter, member of the unprotected headers map, shall be a CBOR array whose elements contain CBOR values that are not signed by the CB-AdES signature.

NOTE 1: The rationale for this is as follows: to allow validating CB-AdES signatures long after they have been created (even after the expiration of the signing certificate, the revocation of some certificate, or even the break of some cryptographic algorithm), all the required material for validating them, and time-stamp tokens that time-stamp all the components of the signatures themselves, are added as unprotected header parameters (they are added after that the digital signature has been created). These time-stamps are called archive time-stamps. As time goes by, it is necessary to add new archive time-stamps for countering new expirations of certificates or new breaks of cryptographic algorithms, for instance. Each newly incorporated archive time-stamp time-stamps all the components in the CB-AdES signature, including the components of the unprotected headers map: the input to its message imprint computation is the result of concatenating all these components. The validation of these augmented signatures requires the verification of the successive archive time-stamps. These verifications require that the message imprint of each archive time-stamp is known without any ambiguity. As CBOR maps do not preserve the order, placing the validation material and the successive archive time-stamps as members of the unprotected headers map would prevent the proper computation of their message imprints. As CBOR arrays define an order, the present document defines the `uHeaders` parameter as a CBOR array to be the container of all the unsigned header parameters that are added to CB-AdES signature after its generation for achieving long term digital signatures. These unsigned header parameters are placed in the CBOR array in their order of incorporation. In this way, for each new archive time-stamp to be incorporated to the CB-AdES signature, the input to its message imprint computation is built by concatenating, among other things, all the elements present in the `uHeaders` parameter in the order of appearance within this CBOR array. ETSI EN 319 102-1 [i.3] defines an algorithm for validating digital signatures augmented with archive time-stamps.

NOTE 2: As it has been specified in clause 4 of the present document `uHeaders` header parameter is incorporated in the unprotected headers map specified in clause 3 of IETF RFC 9052 [2]. Consequently, all its elements will also be unprotected. The `uHeaders` header parameter plays in CB-AdES signatures the same role as the `etsiU` header parameter in JAdES signatures (which is specified in clause 5.3.1 of ETSI TS 119 182-1 [9].

The `uHeaders` header parameter shall contain CBOR values that qualify the CB-AdES signature itself, or the signer, or the COSE Payload.

New unsigned attributes shall always be added at the end of the `uHeaders` header parameter, which is a CBOR array.

NOTE 3: This implies that the order in the `uHeaders` header parameter reflects the order in the generation of the CBOR elements.

The unsigned attributes shall be encapsulated in CBOR byte strings before being placed within the `uHeaders` header parameter (CBOR-bstr-wrapped).

The present document specifies:

1) A CBOR map (`sigPSt`) containing details for facilitating access to a signature policy document, in clause 5.3.2.

2) CBOR objects containing details for a counter signature of the CB-AdES signature itself, in clause 5.1.6.

3) A CBOR map (`sigTst`) containing a time-stamp token on the COSE signature value, in clause 5.3.3.

4) A CBOR map (`valData`) containing certificates and validation data required for validating the signature, in clause 5.3.4.

5) A CBOR map (`arcTst`) containing one or more time-stamp tokens on all the components of the CB-AdES signature, in clause 5.3.5.

6) A CBOR map (`refs`) containing references to certificates and validation data required for validating the signature, in clause A.1.1.

7) A CBOR map (`sigRTst`) containing a time-stamp token on the references to the validation material and the COSE Signature Value, in clause A.1.2.1.

8) A CBOR map (`rfsTst`) containing a time-stamp token on the references to the validation material, in clause A.1.2.2.

All the CBOR objects listed above shall be placed within the `uHeaders` header parameter if they are incorporated into the CB-AdES signature.

The `uHeaders` CBOR array shall be assigned an identifying tag. Additionally, each unsigned attribute shall be assigned a label.

Table 8 below defines the labels associated to each unsigned attribute and the tag identifying the `uHeaders` CBOR array specified in clause 5.3.

Table 8: Tags for signed header parameters specified in the present document

unsigned attributes type name	uHeaders CBOR array tag and unsigned attributes labels
uHeaders	268
sigTst	1
valData	2
arcTst	3
refs	4
sigRTst	5
rfsTst	6
sigPSt	7

The following CDDL rules assign the integer values of the labels of the unsigned header parameters to identifiers that will be used throughout the rest of the present document for making the CDDL rules easier to read:

```
uHeaders_l = 268
sigTst_l = 1
valData_l = 2
arcTst_l = 3
refs_l = 4
sigRTst_l = 5
rfsTst_l = 6
sigPSt_l = 7
```

NOTE 4: While integer labels have been assigned to all the header parameters in the present document, it is allowed to incorporate header parameters defined elsewhere with labels whose values are text strings.

Syntax

Below follows the CDDL definition of the `uHeaders` header parameter:

```
uHeaders =  [+bstr .cbor UHeaderInstance]   ;an array of CBOR byte strings each on encapsulating
                                                ;one instance of UHeaderInstance


UHeaderInstance = {
    sigTst_l => sigTst //           ;Signature time-stamp OR
    valData_l => valData //         ;Validation data (certificate values and revocation data) OR
    arcTst_l => arcTst //           ;Archive-time-stamp OR
    refs_l => refs //               ;References to certificates and revocation data OR
    sigRTst_l => sigRTst //         ;Signature and references time-stamp OR
    rfsTst_l => rfsTst //           ;References only time-stamp OR
    sigPSt_l => sigPSt //           ; Signature Policy Store OR
    11 => COSE_CounterSignature /   ;full counter signature as specified in IETF RFC 9338
            [+COSE_CounterSignature] //
    12 => COSE_CounterSignature0 // ;abbreviated counter signature as specified in IETF RFC 9338
    33 => bstr / [2*certs:bstr] // ;x5chain for the signing certificate
    *label => value             ;other additional unsigned attributes not specified in the
                                    ;present document
}
```

The `uHeaders` header parameter shall be a non-empty array.

In CB-AdES signatures supported by a `COSE_Sign` structure, this header parameter shall be placed at the signer layer.

The `uHeaders` header parameter shall be incorporated as member of the unprotected header map of the CB-AdES signature.

NOTE 5: The second element of the CBOR array forming the COSE object structure, is the place reserved by IETF RFC 9052 [2] for unsigned header parameters in COSE Signatures. Clause 3.2 of IETF RFC 9052 [2] leaves its content open. The present document suitably profiles its contents.

The `uHeaders` header parameter should be the only header parameter incorporated to the unprotected headers map. Any CBOR value that is not specified in the present document should be incorporated as an element of the `uHeaders` header parameter.

NOTE 6: Adding these components into the `uHeaders` header parameter allows to properly secure them in the long-term using `arcTst`.

5.3.2 The `sigPSt` CBOR map

Semantics

The `sigPSt` CBOR map shall contain either:

- the signature policy document which is referenced in the `sigPId` CBOR map so that the signature policy document can be used for offline and long-term validation; or
- a URI referencing a local store where the signature policy document can be retrieved.

Syntax

Below follows the CDDL definition of the `sigPSt` header parameter:

```
sigPSt = {
    1 => DocOrLocalURI, ;docOrLocalUri: either the signature policy document itself or a URI
                        ;referene to it
    ?2 => oId           ;spDSpec: identifier of the technical specification defining the syntax
                        ;of the signature policy
}

DocOrLocalURI = {
    1 => bstr //        ;sigPolDoc: The signature policy document itself
    2 => #6.32(tstr),   ;sigPolLocalURI: a local URI to the signature policy document
}
```

The `sigPolDoc` member shall contain the signature policy document encapsulated within a CBOR byte string.

The `sigPolLocalURI` member shall have as value the URI pointing to a local store where the present document can be retrieved.

NOTE 1: Contrary to the `spURI`, the `sigPolLocalURI` points to a local file.

The `spDSpec` member shall identify the technical specification that defines the syntax used for producing the signature policy document.

NOTE 2: It is the responsibility of the entity incorporating the signature policy to the signature-policy-store to make sure that the correct document is securely stored.

NOTE 3: Being unsigned, the `sigPSt` is not protected by the digital signature. If the `sigPId` signed attribute is incorporated into the signature and contains the `digAlgVal` member with the digest value of the signature policy document, any alteration of the signature policy document present within `sigPSt` or within a local store, would be detected by the failure of the digests comparison.

Table 9 shows the values assigned to each of the keys in the maps specified in the present clause.

Table 9: Values of keys in maps or groups specified in the present clause

Name	Label in map
`docOrLocalUri (in sigPst)`	1
`spDSpec (in sigPst)`	2
`sigPolDoc (in DocOrLocalURI group)`	1
`sigPolLocalURI(in DocOrLocalURI group)`	2

5.3.3 The sigTst CBOR map

Semantics

The sigTst CBOR map shall encapsulate one or more electronic time-stamps time-stamping the COSE signature value.

Syntax

Below follows the CDDL definition of the sigTst header parameter:

```
sigTst = tstContainer
```

The input of the message imprint computation for the time-stamp tokens encapsulated by sigTst CBOR map shall be the COSE signature value present within the CB-AdES signature.

NOTE: This is the same as the content encapsulated within the signature CBOR byte string member of instances of COSE_Signature type specified in IETF RFC 9052 [2], clause 4.1.

5.3.4 The valData CBOR map

Semantics

The valData CBOR map shall contain the certificates identified in 1) below, or the revocation data identified in 2) below, or both of them:

1) Certificate values that are used for validating any digital signature present within any component of the CB-AdES signature regardless the objects that they are signing (these can be, for instance, the COSE signature value within the signature component itself, any counter signature of the CB-AdES signature, or the digital signatures within any time-stamp token, attribute certificate, signed assertion, OCSP response, or CRL, or any other digital signature), without any restrictions.

2) Revocation value(s) of the certificate(s) supporting any signature present within any component of the CB-AdES signature mentioned in the previous bullet.

NOTE 1: This CBOR map allows mimicking, within CB-AdES, features already incorporated in PAdES and CAdES, namely: an unsigned component whose purpose is to contain certificates and validation material that can be used for validating any signature present within CB-AdES signatures, regardless what these signatures are signing.

NOTE 2: This CBOR map also allows for properly dealing with situations where different creation/validation/augmentation signature policies can be used. They, for instance, may establish different requirements on acceptable freshness of revocation material, and also allow different certificate paths. Therefore, a certain set of revocation data fully acceptable for a certain policy A, may be unacceptable, from the point of view of its freshness, for another policy B. Also a verifier can accept a different certificate path. This unsigned component allows, for instance, including within a CB-AdES signature a set of revocation data whose freshness is acceptable for this last policy B, making the signature valid under both policies.

Syntax

Below follows the CDDL definition of the valData CBOR map:

```
valData = {
    ? 1 => xVals,   ;xVals: DER-encoded encapsulated X.509 certificates or other
                    ;encapsulated certificates
    ? 2 => rVals    ;rVals: validation material
}

xVals = [
    +X509OrOther
]

X509OrOther = {
    1 => pkiOb, //  ; x509Cert: DER-encoded X.509 certificate encapsulated in an instance
                    ;of pkiOb type
```

```
    2 => pkiOb      ; otherCert: Other type of certificate encoded and encapsulated in an instance
                    ;of pkiOb type
}

rVals = {
    ? 1 => [+pkiOb],    ;crlVals: array of CRLs encapsulated in an instance of pkiOb type
    ? 2 => [+pkiOb],    ;ocspVals: array of DER-encoded OCSPResponse encapsulated in an instance
                            ;of pkiOb type
    ? 3 => [+pkiOb] ;otherVals: other revocation values encoded and encapsulated in
                            ; an instance of pkiOb type
}
```

CB-AdES signatures shall not incorporate empty `valData` maps.

`xVals` map shall have at least one member.

An `x509Cert` item shall contain one DER-encoded X.509 certificate encapsulated within an instance of `pkiOb` type.

An `otherCert` item is a placeholder for potential future new formats of certificates.

`rVals` map shall have at least one member.

`crlVals` member shall be a non-empty array of DER-encoded X.509 CRLs [13].

Each element of `crlVals` array shall contain one DER-encoded X.509 CRL [13] encapsulated in a CBOR byte string.

If the validation data contain one or more Delta CRLs, the `crlVals` member shall contain the set of CRLs required to provide complete revocation lists.

`ocspVals` member shall be a non-empty array of DER-encoded OCSP responses [14].

Each item of `ocspVals` array shall contain a DER-encoded instance `OCSPResponse` defined in IETF RFC 6960 [14], clause 4.2.1.

The `otherVals` member provides a placeholder for other revocation information that can be used in the future. Their semantics and syntax are outside the scope of the present document.

Table 10 shows the values assigned to each of the keys in the maps specified in the present clause.

Table 10: Values of keys in maps or groups specified in the present clause

Name	Label in map
xVals (in valData)	1
rVals (in valData)	2
X509Cert (in X509OrOther)	1
otherCert (in X509OrOther)	2
crlVals (in rVals)	1
ocspVals (in rVals)	2
otherVals (in rVals)	3

5.3.5 The `arcTst` CBOR map

5.3.5.1 Semantics and syntax

Semantics

The `arcTst` CBOR map shall encapsulate electronic time-stamps computed on the COSE Payload, the protected headers map or maps (depending on the used signature structure), the COSE signature value, the externally supplied data, when present, and the `uHeaders` CBOR array within the unprotected headers map at the time of generating each electronic time-stamp.

NOTE 1: The purpose of this CBOR map is to tackle the long-term availability and integrity of the validation material.

NOTE 2: As it has been anticipated in clause 4 any header parameter different than `uHeaders` CBOR array present within the unprotected headers map is not protected by the time-stamps encapsulated by this CBOR map.

Syntax

Below follows the CDDL definition of the `arcTst` CBOR map:

```
arcTst = tstContainer
```

If the CB-AdES signature incorporates a counter signature element, all the required material for conducting the validation of the counter signature shall be incorporated into the CB-AdES signature before generating the first `arcTst` CBOR map. This may be done within the counter signature itself or within the containers available within the counter-signed CB-AdES signature.

The contents of the counter signature element should not be changed, once it has been time-stamped by the `arcTst`.

NOTE 3: If a counter signature element is time-stamped by the `arcTst`, any subsequent change of its contents (by addition of unsigned CBOR values if the counter signature is a CB-AdES signature, for instance) would make the validation of the `arcTst` fail.

The `tstContainer` member shall be as specified in clause 5.4.3.3 of the present document.

5.3.5.2 Generation and incorporation of `arcTst`

The steps listed below shall be performed for augmenting a CB-AdES signature by incorporation of a new `arcTst` CBOR map:

1) If the CB-AdES signature misses certificates and/or revocation data required for validating the signed objects present in the CB-AdES signature, then these missing certificates and/or revocation data shall be encapsulated within a new `valData` CBOR map. This new `valData` CBOR map shall be incorporated to the CB-AdES signature before generating the electronic time-stamp(s) to be encapsulated by the `arcTst` CBOR map.

2) Compute the message imprint for the new archive time-stamp token(s), as indicated in clause 5.3.5.3.

3) Request as many archive time-stamp token(s) as required to the corresponding time-stamp token Service Providers.

4) Build a new `arcTst` CBOR map, encapsulating the time-stamp token(s) issued in the previous step and wrap it.

5) Wrap the `arcTst` CBOR map in a CBOR byte string and incorporate it as the last element in the `uHeaders` CBOR array.

5.3.5.3 Computation of message-imprint for `arcTst`

For computing the input to the message imprint computation, indicated in step 2) in clause 5.3.5.2, the steps listed below shall be performed:

1) Initialize an empty CBOR array.

2) Add a context text string, whose value shall be either:

   - "Signature", if the CB-AdES signature is built on the `COSE_Sign` structure defined in IETF RFC 9052 [2]; or

   - "Signature1", if the CB-AdES signature is built on the `COSE_Sign1` structure defined in IETF RFC 9052 [2]; or

   - the context text string corresponding to the structure of the CB-AdES signature if it is a counter signature, as specified in clause 3.3 of IETF RFC 9338 [6].

3) Add the protected header from the body layer, encapsulated in a CBOR byte string. If the body layer does not have the protected header, add a zero-length CBOR byte string.

4) If the CB-AdES signature is built on the `COSE_Sign` structure, then:

   - If the protected header map is present in the signer layer, add the protected header from the signer layer, encapsulated in a CBOR byte string.

   - Else if the protected header map is absent in the signer layer, add a zero-length CBOR byte string.

5) Add the externally supplied data from the application, encapsulated in a CBOR byte string. If no data is externally supplied to the application, add a zero-length CBOR byte string.

6) If the `sigD` header parameter is absent, then:

   - If the payload field is present, then add the CBOR byte string of the payload field.

   - Else if the payload field is absent (COSE Payload is detached, and not explicitly referenced by the `sigD` header parameter), then retrieve the bytes of the COSE Payload and add them encapsulated in a CBOR byte string.

NOTE 1: It is out of the scope of the present document to specify how the bytes of the unreferenced detached COSE Payload are retrieved.

7) If the `sigD` header parameter is present, retrieve the bytes resulting from processing the contents of its `pars` member as specified in clause 5.2.8.2.2 of the present document, concatenate them, encapsulate them in a CBOR byte string, and add this CBOR byte string.

NOTE 2: The rationale for applying the processing specified in clause 5.2.8.2.2 of the present document to the case of the mechanism identified by "`http://uri.etsi.org/19152/ObjectIdByURIHash`" is the fact that this is an indirect signing mechanism, i.e. based on signing digest values of data objects, instead of the data objects themselves. Time-stamping not the digest values but the retrieved data objects, protects against future weaknesses of the digest algorithms used in `sigD`.

8) If the CB-AdES signature is built on a version 2 counter signature defined in IETF RFC 9338 [6], add `other_fields` CBOR array, as defined in clause 3.3 of IETF RFC 9338 [6].

9) Add the CBOR byte string in the `signature` component.

10) If the CB-AdES signature is built on the `COSE_Sign` structure, take the elements in the `uHeaders` header parameter from the signer layer in the order that they appear within `uHeaders`, and add them to the CBOR array. If the signer layer does not have the `uHeaders` header parameter, add a zero-length CBOR byte string.

11) Else if the CB-AdES signature is built on the `COSE_Sign1` structure, take the elements in the `uHeaders` header parameter from the body layer in the order that they appear within `uHeaders` and add them to the CBOR array. If the body layer does not have the `uHeaders` header parameter, add a zero-length CBOR byte string.

12) Encode the generated CBOR array in a CBOR byte string.

The CBOR byte string resulting of step 11) shall be the input to the message imprint computation.

As a consequence of the previous process, **for validating a time-stamp token placed within one specific `arcTst` CBOR map present in a CB-AdES signature** as specified in the first paragraph of this clause, its message imprint shall be built as indicated in steps 1) to 12), BUT replacing 10) and 11) with the following ones:

10) If the CB-AdES signature is built on the `COSE_Sign` structure, take the elements in the `uHeaders` header parameter from the signer layer **that precede (appear BEFORE) the `arcTst` CBOR map that contains the time-stamp token that is being validated**, in the order they appear within `uHeaders`, and add them to the CBOR array. If `uHeaders` header parameter is not present, add a zero-length CBOR byte string.

11) Else if the CB-AdES signature is built on the `COSE_Sign1` structure, take the elements in the `uHeaders` header parameter from the body layer **that precede (appear BEFORE) the `arcTst` CBOR map that contains the time-stamp token that is being validated**, in the order they appear within `uHeaders`, and add them to the CBOR array. If `uHeaders` header parameter is not present, add a zero-length CBOR byte string.

5.4 Generally useful syntax

5.4.1 The obId data type

Semantics

Instances of obId data type shall contain a unique and permanent identifier of one data object.

Instances of obId data type may contain a textual description of the nature of the data object qualified by the instance of the obId data type.

Instances of obId data type may contain a number of references to documents where additional information about the nature of the data object identified, can be found.

Syntax

Below follows the CDDL definition of the obId data type:

```
oId = {
    1 => #6.32(tstr),       ;id: the URI that is the object identifier
    ? 2 => tstr,            ;desc: a textual description of the identified object
    ? 3 => [
        +ref: #6.32(tstr)   ;docRefs: an array of URIs pointing to documents
                            ;specifying the identified object
      ]
}
```

The id member shall contain a permanent identifier. Once the identifier is assigned, it shall not be re-assigned again.

The value of the id member shall be an URI. If the identifier of the object is an OID then the value of this member shall be encoded as an URN as specified by the IETF RFC 3061 [7].

If both an OID and a URI exist identifying one object, the URI value should be used in the id member.

The desc member shall contain a short and informal description of the data object identified.

The docRefs member shall contain an arbitrary number of URI values pointing to documents that contain a complete specification of the data object identified.

Table 11 shows the values assigned to each of the keys in the maps specified in the present clause.

Table 11: Values of keys in maps or groups specified in the present clause

Name	Label in map
id	1
desc	2
docRefs	3

5.4.2 The pkiOb data type

Semantics

The pkiOb data type shall be used to incorporate PKI objects, which can be non-CBOR encoded, into the CB-AdES signature.

NOTE: Examples of such PKI objects, include X.509 certificates and revocation lists, OCSP responses, attribute certificates, and electronic time-stamps.

Syntax

Below follows the CDDL definition of the pkiOb type:

```
pkiOb = {
    1 => bstr,          ;val: CBOR byte string encapsulating the encoded PKI object
```

```
    ? 2 => #6.32(tstr), ;encoding: an URI identifying the encoding
    ? 3 => #6.32(tstr)  ;specRef: an URI identifying the specification of the
                        ;encapsulated PKI object
}
```

The `encoding` member's value shall be a URI identifying the encoding used in the original PKI object. The values for the URI shall be one of the values defined in clause 5.1.3 of ETSI EN 319 132-1 [8].

EXAMPLE: Clause 5.1.3 of ETSI EN 319 132-1 [8] defines the URI http://uri.etsi.org/01903/v1.2.2#DER for denoting that the original PKI data were ASN.1 data encoded in DER.

If the `encoding` member is not present, then the contents of `val` member shall be a CBOR byte string encapsulating the DER-encoded ASN.1 data. Otherwise, the contents of `val` member shall be a CBOR byte string encapsulating the PKI object encoded as indicated by the value of `encoding` member.

The `specRef` member shall contain an URI value identifying the specification that defines the encapsulated PKI object.

Table 12 shows the values assigned to each of the keys in the maps specified in the present clause.

Table 12: Values of keys in maps or groups specified in the present clause

Name	Label in map
`val`	1
`encoding`	2
`specRef`	3

5.4.3 Container for electronic time-stamps

5.4.3.1 Introduction

The present document specifies CBOR maps that act as electronic time-stamps containers.

Electronic time-stamps within the aforementioned containers may time-stamp isolated components or concatenations of several components of CB-AdES signatures.

This clause specifies a CBOR type for containers of electronic time-stamps.

5.4.3.2 Containers for electronic time-stamps

Below follows the list of the electronic time-stamps containers that are defined by the present document:

- Container for electronic time-stamps proving that the COSE Payload has been created before certain time instant: `adoTst` (specified in clause 5.2.6 of the present document).
- Container for electronic time-stamps proving that the COSE signature value has been computed before a certain time instant (to protect against repudiation in case of a key compromise): `sigTst` (specified in clause 5.3.3 of the present document).
- Container for electronic time-stamps time-stamping the signature and validation data values, for providing long term CB-AdES signatures: `arcTst` (specified in clause 5.3.5 of the present document).
- Containers for electronic time-stamps on components that contain references to validation data, namely: `rfsTst` and `sigRTst`. (specified in clause A.1.2 of the present document).

5.4.3.3 The `tstContainer` type

Semantics

The `tstContainer` type shall:

- allow encapsulating IETF RFC 3161 [12] electronic time-stamps as well as electronic time-stamps in other formats;
- provide means for managing electronic time-stamps computed on a concatenation of CB-AdES components (including detached COSE Payload); and
- allow encapsulating more than one electronic time-stamp generated for the same set of CB-AdES components (including detached COSE Payload), each one issued by different TSAs, for instance.

Syntax

Below follows the CDDL definition of the `tstContainer` type:

```
tstContainer = {
    1 => [+TstToken];tstTokens: CBOR array containing one or more time-stamp tokens
}

TstToken = {
    1 : bstr,   ;val: Encoded time-stamp token encapsulated in a CBOR byte string
    ? 2 => tstr,    ;type: String identifying the type of time-stamp token
    ? 3 => #6.32(tstr), ;encoding: URI identifying the type of encoding
    ? 4 => #6.32(tstr)  ;specRef: a URI pointing to the reference where
                        ;the time-stamp token is defined
}
```

The `tstContainer`'s `tstTokens` member shall contain a non-empty array of CBOR maps each one encapsulating one electronic time-stamp token.

The `tstToken`'s `type` member shall identify the type of the time-stamp token. For IETF RFC 3161 [12] time-stamp tokens this member shall not be present.

The `tstToken`'s `encoding` member shall be an URI value and shall identify the encoding used for the time-stamp token. For IETF RFC 3161 [12] time-stamp tokens this member shall not be present.

The `tstToken`'s `specRef` member shall be an URI value and shall identify the technical specification that has defined the used time-stamp token. For IETF RFC 3161 [12] time-stamp tokens this member shall not be present.

Finally the `tstToken`'s `val` member shall contain the encoded electronic time-stamp token itself. For IETF RFC 3161 [12] time-stamp tokens this member shall contain the DER-encoded electronic time-stamp token.

In CB-AdES signatures, the containers of time-stamp tokens time-stamping components within the `uHeaders` unsigned header parameter, implicitly identify what components are time-stamped and how they contribute to the input of the message imprint's computation. No further information in the time-stamp token container is required.

NOTE: This is because all the components of a CB-AdES signature are placed within CB-AdES signature itself.

Table 13 shows the values assigned to each of the keys in the maps specified in the present clause.

Table 13: Values of keys in maps or groups specified in the present clause

Name	Label in map
tstTokens (in tstContainer)	1
val (in TstToken)	1
type (in TstToken)	2
encoding (in TstToken)	3
specRef (in TstToken)	4

6 CB-AdES baseline signatures

6.1 Signature levels

Clause 6 defines four levels of CB-AdES baseline signatures, intended to facilitate interoperability and to encompass the life cycle of CB-AdES signature, namely:

a) B-B level provides requirements for the incorporation of signed header parameters and some unsigned components within the `uHeaders` unsigned header parameter when the signature is generated.

b) B-T level provides requirements for the generation and inclusion, for an existing signature, of a trusted token proving that the signature itself actually existed at a certain date and time.

c) B-LT level provides requirements for the incorporation of all the material required for validating the signature in the signature document. This level aims to tackle the long-term availability of the validation material.

d) B-LTA level provides requirements for the incorporation of electronic time-stamps that allow validation of the signature long time after its generation. This level aims to tackle the long-term availability and integrity of the validation material.

NOTE 1: ETSI TR 119 100 [i.6] provides a description on the life-cycle of a signature and the rationales on which level is suitable in which situation.

NOTE 2: The levels c) to d) are appropriate where the technical validity of signature needs to be preserved for a period of time after signature creation where certificate expiration, revocation and/or algorithm obsolescence is of concern. The specific level applicable depends on the context and use case.

NOTE 3: B-LTA level targets long term availability and integrity of the validation material of digital signatures over long term. The B-LTA level can help to validate the signature beyond many events that limit its validity (for instance, the weakness of used cryptographic algorithms, or expiration of validation data). The use of B-LTA level is considered an appropriate preservation and transmission technique for signed data.

NOTE 4: Conformance to B-LT level, when combined with appropriate additional preservation techniques tackling the long term availability and integrity of the validation material is sufficient to allow validation of the signature long time after its generation. The assessment of the effectiveness of preservation techniques for signed data other than implementing the B-LTA level are out of the scope of the present document. The reader is advised to consider legal instruments in force and/or other standards (for example ETSI TS 101 533-1 [i.9] or IETF RFC 4998 [i.10]) that can indicate other preservation techniques. Annex C defines what needs to be taken into account when using other techniques for long term availability and integrity of validation data and incorporating a new component in the `uHeaders` unsigned header parameter derived from these techniques into the signature.

6.2 General requirements

6.2.1 Algorithm requirements

The algorithms and key lengths used to generate and augment digital signatures should be as specified in ETSI TS 119 312 [19].

NOTE: Cryptographic suites recommendations defined in ETSI TS 119 312 [19] can be superseded by national recommendations.

In addition, MD5 algorithm shall not be used as digest algorithm.

6.2.2 Notation for requirements

The present clause describes the notation used for defining the requirements of the different CB-AdES signature levels.

The requirements on the header parameters and certain other signature's components for each CB-AdES signature level are expressed in Table 14. A row in the table either specifies requirements for a header parameter, other signature's component, or a service.

A service can be provided by different header parameters, by other signature's components, or by other mechanisms (service provision options hereinafter). In these cases, the specification of the requirements for a service is provided by three or more rows. The first row contains the requirements of the service. The requirements for the header parameters, other signature's components, and/or mechanisms used to provide the service are stated in the following rows.

Table 14 contains 8 columns. Below follows a detailed explanation of their meanings and contents:

1) Column "Header parameters/Elements in `uHeaders` unsigned header parameter/Services":

   a) In the case where the cell identifies a Service, the cell content starts with the keyword "Service" followed by the name of the service.

   b) In the case where the header parameter or other signature's component provides a service, this cell contains "SPO" (for Service Provision Option), followed by the name of the header parameter or the other signature's component.

   c) Otherwise, this cell contains the name of the header parameter or the other signature's component.

2) Column "Presence in B-B level": This cell contains the specification of the presence of the header parameter or other signature's component, or the provision of a service, for CB-AdES-B-B signatures.

3) Column "Presence in B-T level": This cell contains the specification of the presence of the header parameter or other signature's component, or the provision of a service, for CB-AdES-B-T signatures.

4) Column "Presence in B-LT level": This cell contains the specification of the presence of the header parameter or other signature's component, or the provision of a service, for CB-AdES-B-LT signatures.

5) Column "Presence in B-LTA level": This cell contains the specification of the presence of the header parameter or other signature's component, or the provision of a service, for CB-AdES-B-LTA signatures. Below follow the values that can appear in columns "Presence in B-B", "Presence in B-T", "Presence in B-LT", and "Presence in B-LTA":

   - "shall be present": means that the header parameter or signature's component shall be incorporated to the signature, and shall be as specified in the document referenced in column "References", further profiled with the additional requirements referenced in column "Requirements", and with the cardinality indicated in column "Cardinality".

   - "shall not be present": means that the header parameter or signature's component shall not be incorporated to the signature.

   - "may be present": means that the header parameter or signature's component may be incorporated to the signature, and shall be as specified in the document referenced in column "References", further profiled with the additional requirements referenced in column "Requirements", and with the cardinality indicated in column "Cardinality".

   - "shall be provided": means that the service identified in the first column of the row shall be provided as further specified in the SPO-related rows. This value only appears in rows that contain requirements for services. It does not appear in rows that contain requirements for header parameters or signature's components.

   - "conditioned presence": means that the incorporation to the signature of the item identified in the first column is conditioned as per the requirements referenced in column "Requirements" and requirements in specifications and clauses referenced by column "References", with the cardinality indicated in column "Cardinality".

   - "*": means that the header parameter or signature's component (service) identified in the first column should not be incorporated to the signature (provided) in the corresponding level. Upper signature levels may specify other requirements.

NOTE: Incorporating an unsigned component within the `uHeaders` header parameter that is marked with a "*" into a signature can lead to cases where a higher level cannot be achieved, except by removing the corresponding component.

6) Column "Cardinality": This cell indicates the cardinality of the header parameter or other signature's component. If the cardinality is the same for all the levels, only the values listed below appear. Otherwise the content specifies the cardinality for each level. See the example at the end of the present clause showing this situation. Below follows the values indicating the cardinality:

   - **0:** The signature shall not incorporate any instance of the header parameter or the signature's component.
   - **1:** The signature shall incorporate exactly one instance of the header parameter or the signature's component.
   - **0 or 1:** The signature shall incorporate zero or one instance of the header parameter or the signature's component.
   - **≥ 0:** The signature shall incorporate zero or more instances of the header parameter or the signature's component.
   - ≥ **1:** The signature shall incorporate one or more instances of the header parameter or the signature's component.

7) Column "References": This shall contain either the number of the clause specifying the header parameter in the present document, or a reference to the document and clause that specifies the other signature's component.

8) Column "Additional requirements and notes": This cell contains numbers referencing notes and/or letters referencing additional requirements on the header parameter or the other signature's component. Both notes and additional requirements are listed in Table 14.

6.3 Requirements on CB-AdES components and services

The four CB-AdES signature levels specified in the present clause shall be built as specified in clause 4 of the present document.

Table 14 shows the presence and cardinality requirements on the signature header parameters, other components, and services indicated in the first column for the four CB-AdES baseline signature levels, namely: CB-AdES-B-B, CB-AdES-B-T, CB-AdES-B-LT, and CB-AdES-B-LTA). Additional requirements are detailed below the table suitably labelled with the letter indicated in the last column.

NOTE 1: CB-AdES-B-B signatures that incorporate only the header parameters and other components that are mandatory in Table 14, and that implement the mandatory requirements, contain the lowest number of header parameters and other components, with the consequent benefits for interoperability.

In CB-AdES baseline signatures the components that act as electronic time-stamps containers shall encapsulate only IETF RFC 3161 [12] updated by IETF RFC 5816 [15] time-stamp tokens.

Any header parameter specified in IETF RFC 9052 [2] or IETF RFC 9360 [3], and not further profiled in clause 5.1, may be present (cardinality of 0 or 1) in the four levels defined in Table 14.

Table 14: Requirements for CB-AdES-B-B, CB-AdES-B-T, CB-AdES-B-LT and CB-AdES-B-LTA signatures

Header parameters/Elements in uHeaders unsigned header parameter/Services	Presence in B-B level	Presence in B-T level	Presence in B-LT level	Presence in B-LTA level	Cardinality	References	Additional requirements and notes
alg	shall be present	shall be present	shall be present	shall be present	1	Clause 5.1.2	
content type	conditioned presence	conditioned presence	conditioned presence	conditioned presence	0 or 1	Clause 5.1.3	2
kid	may be present	may be present	may be present	may be present	0 or 1	Clause 5.1.4	
x5u	may be present	may be present	may be present	may be present	0 or 1	Clause 5.1.5	
x5chain	Conditioned presence	Conditioned presence	Conditioned presence	Conditioned presence	0 or 1	Clause 5.1.8	3
crit	Conditioned presence	Conditioned presence	Conditioned presence	Conditioned presence	0 or 1	Clause 5.1.10	4
CWT Claims enclosing the iat	shall be present	shall be present	shall be present	shall be present	1	Clause 5.1.9	a
x5t	conditioned presence	conditioned presence	conditioned presence	conditioned presence	0 or 1	Clause 5.1.7	3
x5ts	conditioned presence	conditioned presence	conditioned presence	conditioned presence	0 or 1	Clause 5.2.2	3
sigD	may be present	may be present	may be present	may be present	0 or 1	Clause 5.2.8	
srAts	may be present	may be present	may be present	may be present	0 or 1	Clause 5.2.5	
srCms	may be present	may be present	may be present	may be present	0 or 1	Clause 5.2.3	5
sigPl	may be present	may be present	may be present	may be present	0 or 1	Clause 5.2.4	
sigPId	may be present	may be present	may be present	may be present	0 or 1	Clause 5.2.7	
counter signature	may be present	may be present	may be present	may be present	≥ 0	Clause 5.1.6	
adoTst	may be present	may be present	may be present	may be present	0 or 1	Clause 5.3.2	6
sigPSt	conditioned presence	conditioned presence	conditioned presence	conditioned presence	0 or 1	Clause 5.3.2	b
sigTst	*	shall be present	shall be present	shall be present	B-B: ≥ 0 B-T, B-LT, B-LTA: ≥ 1 B-LT, B-LTA: 0 B-LT, B-LTA: 0	Clause 5.3.3	c, d 7
valData	*	*	conditioned presence	conditioned presence	≥ 0	Clause 5.3.4	e, f
refs	*	*	shall not be present	shall not be present	B-B, B-T: ≥ 0 B-LT, B-LTA: 0	Clause A.1.1	g
sigRTst	*	*	shall not be present	shall not be present	B-B, B-T: ≥ 0 B-LT, B-LTA: 0	Clause A.1.2.1	
rfsTst	*	*	shall not be present	shall not be present	B-B, B-T: ≥ 0 B-LT, B-LTA: 0	Clause A.1.2.2	

Header parameters/Elements in `uHeaders` unsigned header parameter/Services	Presence in B-B level	Presence in B-T level	Presence in B-LT level	Presence in B-LTA level	Cardinality	References	Additional requirements and notes
Service: Incorporation of validation data for electronic time-stamps	*	*	shall be provided	shall be provided	-	-	h, i 8
SPO `valData`	*	*	conditioned presence	conditioned presence	≥ 0	Clause 5.3.4	
SPO: certificate and revocation values embedded in the electronic time-stamp itself	*	*	conditioned presence	conditioned presence	≥ 0	-	i
`arcTst`	*	*	*	shall be present	≥ 1	Clause 5.3.5	j, k

Additional requirements:

a) Requirement for `iat`. The generator shall include the claimed UTC time when the signature was generated as content of the `iat` member of the CWT Claims header parameter.

b) Requirement for `sigPSt`. This header parameter may be incorporated into the CB-AdES signature only if the `sigPId` is also incorporated and it contains the `digVal` member with the digest value of the signature policy document. Otherwise, the `sigPSt` shall not be incorporated into the CB-AdES signature.

c) Requirement for `sigTst`. Each `sigTst` shall contain only one electronic time-stamp.

d) Requirement for `sigTst`. The electronic time-stamp encapsulated within the `sigTst` shall be created before the signing certificate has been revoked or has expired.

e) Requirement for `valData`. Duplication of certificate values within the signature should be avoided.

f) Requirement for `valData`. Duplication of revocation values within the signature should be avoided.

g) Requirement for `refs`. The references to certificates (`xRefs`) should not include the `kid` member.

h) Requirement for service "incorporation of validation data for electronic time-stamps". The validation data for electronic time-stamps shall be present within the `valData` or embedded in the electronic time-stamp itself.

i) Requirement for service "incorporation of validation data for electronic time-stamps". The validation data for electronic time-stamps should be included within `valData`.

j) Requirement for `arcTst`. Each `arcTst` may contain more than one electronic time-stamp issued by different TSAs.

k) Requirement for `arcTst`. Before generating and incorporating a new `arcTst`, all the validation material required for validating the CB-AdES signature shall be included. This validation material shall include all the certificates and all certificate status information (like CRLs or OCSP responses) required for:

- validating the signing certificate;
- validating the signing certificate of any counter signature incorporated into the signature;
- validating any attribute certificate or signed assertion present in the signature; and
- validating the signing certificate of any previous electronic time-stamp already incorporated into the signature within any CB-AdES electronic time-stamp container component (including any `arcTst`).

NOTE 2: On `content type`, and `ctys` within `sigD`: see clauses 5.1.3 and 5.2.8.1 of the present document for details of their conditioned presence.

NOTE 3: On `x5chain`, `x5t`, and `x5ts`. Clause 5.2.2 specifies the conditions that decide the presence or absence of protected `x5chain`, `x5t`, and `x5ts` header parameters in a CB-AdES signature.

NOTE 4: On `crit`. Clause 5.1.10 specifies the conditions that decide the presence or absence of the `crit` header parameter in a CB-AdES signature.

NOTE 5: On `srCms`. As the content of `srCms` is a CBOR array, it may contain several commitment types.

NOTE 6: On `adoTst`. This header parameter may contain more than one electronic time-stamp coming from different TSAs.

NOTE 7: On `sigTst`. Several instances of this components can be incorporated into the CB-AdES signature, coming from different TSAs.

NOTE 8: On service "incorporation of validation data for electronic time-stamps": the incorporation of the validation material of the electronic time-stamps ensures that the CB-AdES signature actually contains all the validation material needed.

Annex A (normative): Additional components Specification

A.1 Components for validation data

A.1.1 The `refs` CBOR map

Semantics

The `refs` CBOR map:

1) May contain references to certificate values that are used for validating any digital signature present within any component of the CB-AdES signature regardless the objects that they are signing (these can be, for instance, the COSE signature value within the `signature` component itself, any counter signature of the CB-AdES signature, or the digital signatures within any time-stamp token, attribute certificate, signed assertion, OCSP response, or CRL, or any other digital signature), without any restrictions.

2) Shall not contain the signing certificate of the CB-AdES signature itself.

3) May contain the references to the revocation value(s) of the certificate(s) supporting any signature present within any component of the CB-AdES signature mentioned in the previous bullet.

Syntax

Below follows the CDDL definition of the `refs` CBOR map:

```
refs = {
    ? 1 => xRefs,   ; xRefs: references to certificates
    ? 2 => rRefs    ; rRefs: references to revocation data
}

xRefs = [+CertId]

CertId = {
    1 => COSE_CertHash, ;x5t: the digest algorithm identifier and value
    ? 2 => int / tstr / bstr,   ;kid: optional key identifier
    ? 3 => #6.32 (tstr) ;x5u: URI pointing to X.509 certificate
}

rRefs = {
    ? 1 => [+CRLRef],   ;crlRefs: array of references to CRLs
    ? 2 => [+OCSPRef],  ;ocspRefs: array of references to OCSP responses
    ? 3 => [+any]       ;otherRefs: array of references to OCSP responses
}

CRLRef = {
    1 => DigAlgVal, ;digAlgVal: digest algorithm and value of DER-encoded CRL.
    ? 2 => CRLId    ;crlId: identifier of the CRL
}

CRLId = {
    1 => bstr,  ;issuer: the DER-encoded issuer of the CRL
    2 => #6.0 (tstr),   ;issueTime: the date and time of issuance
    ? 3 => uint,    ;number: the issuance number
    ? 4 => #6.32 (tstr) ;uri: an URI pointing to the CRL (hint)
}

OCSPRef = {
    1 => DigAlgVal, ;digAlgVal: digest algorithm and value of the OCSP response
    2 => OCSPId ;ocspId: identifier of the OCSP response
}

OCSPId = {
    1 => ResponderIdChoice, ;responderChoice: a choice for identifying the responder
    2 => #6.0 (tstr),       ;producedAt: same time as the time indicated by the ProducedAt
                        ;field of the referenced OCSP response
    ? 3 => #6.32 (tstr) ;uri: an URI pointing to the OCSP response (hint)
```

```
}

ResponderIdChoice = (
    1 => bstr //    ;responderIdByName: the name of the responder wrapped in a CBOR byte string
    2 => bstr   ;responderIdByKey: the key of the responder wrapped in a CBOR byte string
)
```

CB-AdES signatures shall not incorporate empty `refs` CBOR maps.

Empty `xRefs` shall not be incorporated.

Within `xRefs`, the `x5t` member (specified in IETF RFC 9360 [3], clause 2) shall identify the digest algorithm and the digest value computed on the DER-encoded certificate.

The content of `kid` member should be a DER-encoded instance of type `IssuerSerial` type defined in IETF RFC 5035 [10] wrapped in a CBOR byte string.

NOTE 1: The information in the `kid` member is only a hint, that can help to identify the certificate whose digest matches the value present in the reference. But the binding information is the digest of the certificate.

The `x5u` member shall provide an indication of where the referenced certificate can be found.

NOTE 2: It is intended that the `x5u` member is used as a hint, as implementations can have alternative ways for retrieving the referenced certificate if it is not found at the referenced place.

Empty `rRefs` shall not be incorporated.

The `crlRefs` member shall contain a non-empty array of references to CRLs.

Each item within the `CRLRefs` array shall contain one reference to one CRL.

The `digAlgVal` member within the `crlRefs` array shall contain indication of a digest algorithm, and the digest value of the DER-encoded referenced CRL wrapped in a CBOR byte string.

The `crlId` member needs not to be present if the referenced CRL can be inferred from other information.

The `crlId` member of the items within the `crlRefs` array shall include the name issuer in its `issuer` member.

The `crlId` member of the items within the `crlRefs` array shall include the time when the CRL was issued in its `issueTime` member.

The `crlId` member of the items within the `crlRefs` array may include the number of the CRL in its `number` member.

NOTE 3: The `number` member is an optional hint helping to get the CRL whose digest matches the value present in the reference.

The `crlId`'s `uri` member shall indicate one place where the referenced CRL can be found.

NOTE 4: It is intended that this component be used as a hint, as implementations can have alternative ways for retrieving the referenced CRL if it is not found at the referenced place.

If one or more of the identified CRLs are a Delta CRL, this component shall include references to the set of CRLs required to provide complete revocation lists.

The `ocspRefs` member shall contain a non-empty array of references to OCSP responses.

Each item within the `ocspRefs` array shall contain one reference to one OCSP response.

The `ocspId` member of the items within the `ocspRefs` array shall include an identifier of the responder wrapped in a CBOR byte string. If the identifier is the digest of the server's public key computed as mandated in IETF RFC 6960 [14], the member `responderIdByKey` shall be present. If the identifier is the DER-encoded name of the responder, the member `responderIdByName` shall be present.

If the responder is identified by the digest of the server's public key computed as mandated in IETF RFC 6960 [14], then the base64 [17] encoding of the DER-encoded of `byKey` field specified in IETF RFC 6960 [14] shall appear within the `responderID`'s `byKey` member.

The `ocspId` member of the items within the `ocspRefs` array shall include the generation time of the OCSP response in its `producedAt` member.

The value in `ocspId`'s `producedAt` member shall indicate the same time as the time indicated by the `ProducedAt` field of the referenced OCSP response.

The `ocspId`'s `uri` member shall indicate one place where the referenced OCSP response can be found.

NOTE 5: This value is not the address where the OCSP service can be reached. In addition to that, it is intended that this component be used as a hint, as implementations can have alternative ways for retrieving the referenced OCSP response if it is not found at the referenced place.

The `digAlgVal` member within the `ocspRefs` array shall contain indication of a digest algorithm, and the digest value of the DER-encoded `OCSPResponse` field defined in IETF RFC 6960 [14], wrapped in a CBOR byte string.

References to alternative forms of validation data may be included in this component making use of the `otherRefs` member, a sequence whose items may contain any kind of information. Their semantics and syntax are outside the scope of the present document.

If at least one of the following: `valData` or the `arcTst`, is incorporated into the signature, all the certificates and validation data referenced in `refs` shall be present elsewhere in the signature.

Table A.1 shows the values assigned to each of the keys in the maps specified in the present clause.

Table A.1: Values of keys in maps or groups specified in the present clause

Name	Label in map
xRefs (in refs)	1
rRefs (in refs)	2
x5t (in CertId)	1
kid (in CertId)	2
x5u (in CertId)	3
crlRefs (in rRefs)	1
ocspRefs (in rRefs)	2
otherRefs (in rRefs)	3
digAlgVal(in CRLRef)	1
crlId (in CRLRef)	2
issuer (in CRLId)	1
issueTime (in CRLId)	2
number (in CRLId)	3
uri (in CRLId)	4
digAlgVal (in OCSPRef)	1
ocspId (in OCSPRef)	2
uri (in OCSPId map)	3
responderChoice (in OCSPId)	1
producedAt (in OCSPId)	2
responderIdByName (in ResponderIdChoice)	1
responderIdByKey (in ResponderIdChoice)	2

A.1.2 Time-stamps on references to validation data

A.1.2.1 The `sigRTst` CBOR map

A.1.2.1.1 General

Semantics

The `sigRTst` CBOR map shall encapsulate electronic time-stamps on the COSE signature value, the signature time-stamp, if present, and the CB-AdES components containing references to validation data.

Syntax

Below follows the CDDL definition of the `sigRTst` CBOR map:

```
sigRTst = tstContainer
```

This CBOR map shall contain an electronic time-stamp that time-stamps the member encapsulating the COSE signature value, and the following components when they are present: `sigTst`, and `refs`.

If the component `refs` is not present, the `sigRTst` CBOR map shall not be generated.

A.1.2.1.2 Computation of the message imprint

For computing the input to the message imprint, the following steps listed below shall be performed:

1) Initialize an empty CBOR array.

2) Add the CBOR byte string in the `signature` component.

3) If the CB-AdES signature is built on the `COSE_Sign` structure, take those elements in the `uHeaders` header parameter from the signer layer in the order that they appear within `uHeaders`, and add them to the CBOR array:

   - `sigTst` if it is present;
   - `refs` if it is present.

   If the signer layer does not have any of those `uHeaders` header parameters, add a zero-length CBOR byte string.

4) If the CB-AdES signature is built on the `COSE_Sign1` structure, take those elements in the `uHeaders` header parameter from the body layer in the order that they appear within `uHeaders`, and add them to the CBOR array:

   - `sigTst` if it is present;
   - `refs` if it is present.

   If the signer layer does not have any of those `uHeaders` header parameters, add a zero-length CBOR byte string.

5) Encode the generated CBOR array in a CBOR byte string.

A.1.2.2 The `rfsTst` CBOR map

A.1.2.2.1 Semantics and syntax

Semantics

The `rfsTst` CBOR map shall encapsulate electronic time-stamps on the signature time-stamp, if present, and the CB-AdES components containing references to validation data.

Syntax

Below follows the CDDL definition of the `rfsTst` CBOR map:

```
rfsTst = tstContainer
```

This CBOR map shall contain an electronic time-stamp that time-stamps the member encapsulating the following components when they are present: `sigTst`, and `refs`.

If the component `refs` is not present, the `rfsTst` CBOR map shall not be generated.

A.1.2.2.2 Computation of the message imprint

For computing the input to the message imprint, the following steps listed below shall be performed:

1) Initialize an empty CBOR array.

2) If the CB-AdES signature is built on the `COSE_Sign` structure, take those elements in the `uHeaders` header parameter from the signer layer in the order that they appear within `uHeaders`, and add them to the CBOR array:

   - `sigTst` if it is present;
   - `refs` if it is present.

   If the signer layer does not have any of those `uHeaders` header parameters, add a zero-length CBOR byte string.

3) If the CB-AdES signature is built on the `COSE_Sign1` structure, take those elements in the `uHeaders` header parameter from the body layer in the order that they appear within `uHeaders`, and add them to the CBOR array:

   - `sigTst` if it is present;
   - `refs` if it is present.

   If the signer layer does not have any of those `uHeaders` header parameters, add a zero-length CBOR byte string.

4) Encode the generated CBOR array in a CBOR byte string.

Annex B (informative): IANA Considerations

The present document registers the following header parameters in the IANA "CBOR Object Signing and Encryption (COSE) Header Parameters" registry (https://www.iana.org/assignments/cose/cose.xhtml#header-parameters) established by IETF RFC 9052 [2].

NOTE 1: Header parameters whose "Header Parameter Usage Locations" are COSE/CB-AdES are considered to be Header Parameters that can be present in non-CB-AdES COSE signature, for meeting certain (business, regulatory, etc.) requirements.

NOTE 2: Header parameters whose "Header Parameter Usage Locations" are CB-AdES are considered to be Header Parameters that are used only by CB-AdES signatures for meeting the requirements associated to achieve long-term signatures, i.e. signatures that guarantee the long-term availability and integrity of their validation material, providing therefore mechanisms for being properly validated long after their generation. This capability requires the usage of certain Header Parameters (`uHeaders`, `arcTst` and others) which convert the COSE signature in a CB-AdES signature.

Registry Contents:

- Name: `x5ts`
- Label: `261`
- Value Type: `array of COSE_CertHash`
- Description: CBOR array of instances of `COSE_CertHash`
- Reference: clause 5.2.2 of the present document

- Name: `srCms`
- Label: `262`
- Value Type: `array of SrCm`
- Description: set of commitments and optional commitments qualifiers
- Reference: clause 5.2.3 of the present document

- Name: `sigPl`
- Label: `263`
- Value Type: `map`
- Description: CBOR map for indicating the location where the signature was generated. It may contain an indication of the country, the locality, the region, a box number in a post office, the postal code, and the street address
- Reference: clause 5.2.4 of the present document

- Name: `srAts`
- Label: `264`
- Value Type: `map`

- Description: CBOR map that may contain: an array of attributes that the signer claims to be in possession of, an array of attribute certificates (X.509 attribute certificates or other) issued to the signer, an array of signed assertions issued by a third party to the signer, or any combination of the three aforementioned CBOR arrays
- Reference: clause 5.2.5 of the present document

- Name: `adoTst`
- Label: `265`
- Value Type: `map`
- Description: CBOR map that encapsulates one or more electronic time-stamps, generated before the signature production, and whose message imprint computation input is the COSE Payload of the CB-AdES signature
- Reference: clause 5.2.6 of the present document

- Name: `sigPId`
- Label: `266`
- Value Type: `map`
- Description: CBOR map that identifies a certain signature policy and may contain the digest of the document defining this signature policy
- Reference: clause 5.2.7 of the present document

- Name: `sigD`
- Label: `267`
- Value Type: `map`
- Description: CBOR map that references data objects that are detached from the CB-AdES signature and that are collectively signed
- Reference: clause 5.2.8 of the present document

- Name: `uHeaders`
- Label: `268`
- Value Type: `[+bstr]`
- Description: CBOR array that contains a number of CBOR elements that are placed within the array in the order they are incorporated into the CB-AdES signature
- Reference: clause 5.3.1 of the present document

Annex C (informative): URIs defined for commitment type

At the time of publication of the present document, the following commitment types identifiers have been defined:

- Proof of origin: it indicates that the signer recognizes to have created, approved and sent the signed data.

  Object identifier: id-cti-ets-proofOfOrigin OBJECT IDENTIFIER ::= { iso(1) member-body(2)us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) cti(6) 1}

  URI: http://uri.etsi.org/01903/v1.2.2#ProofOfOrigin.

- Proof of receipt: it indicates that signer recognizes to have received the content of the signed data.

  Object identifier: id-cti-ets-proofOfReceipt OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) cti(6) 2}.

  URI: http://uri.etsi.org/01903/v1.2.2#ProofOfReceipt.

- Proof of delivery: it indicates that the TSP providing that indication has delivered a signed data in a local store accessible to the recipient of the signed data.

  Object identifier: id-cti-ets-proofOfDelivery OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) cti(6) 3}.

  URI: http://uri.etsi.org/01903/v1.2.2#ProofOfDelivery.

- Proof of sender: indicates that the entity providing that indication has sent the signed data (but not necessarily created it).

  Object identifier: id-cti-ets-proofOfOrigin OBJECT IDENTIFIER ::= { iso(1) member-body(2)us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) cti(6) 4}.

  URI: http://uri.etsi.org/01903/v1.2.2#ProofOfSender.

- Proof of approval: it indicates that the signer has approved the content of the signed data.

  Object identifier: id-cti-ets-proofOfOrigin OBJECT IDENTIFIER ::= { iso(1) member-body(2)us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) cti(6) 5}.

  URI: http://uri.etsi.org/01903/v1.2.2#ProofOfApproval.

- Proof of creation: it indicates that the signer has created the signed data (but not necessarily approved, nor sent it).

  Object identifier: id-cti-ets-proofOfOrigin OBJECT IDENTIFIER ::= { iso(1) member-body(2)us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) cti(6) 6}.

  URI: http://uri.etsi.org/01903/v1.2.2#ProofOfCreation.

Annex D (informative): Correspondence between JAdES tags and CB-AdES tags

D.1 Correspondence between JAdES components tags and CB-AdES component tags

Table D.1 shows the correspondence between the tags of JAdES components and the names given to the CB-AdES components.

Table D.1: Correspondence between JAdES and CB-AdES tags

JAdES tag	CB-AdES names
etsiU	uHeaders
iat	iat
sigX5ts	x5ts
crit	crit
sigPId	sigPId
sigPl	sigPl
srAts	srAts
cty	content type
adoTst	adoTst
srCms	srCms
cSig	counter signature
sigPSt	sigPSt
sigTst	sigTst
oId	obId
pkiObj	pkiOb
arcTst	arcTst
rfsTst	rfsTst
sigRTst	sigRTst
anyValData	valData
NO equivalent component	refs
xVals	NO equivalent component
rVals	NO equivalent component
axVals	NO equivalent component
arVals	NO equivalent component
tstVd	NO equivalent component
xRefs	NO equivalent component
rRefs	NO equivalent component
axRefs	NO equivalent component
arRefs	NO equivalent component

Annex E (normative): Alternative mechanisms for long term availability and integrity of validation data

There may be mechanisms to achieve long-term availability and integrity of validation data different from the ones described in clause 5.3.5 of the present document.

If such a mechanism is incorporated using an unsigned component into the signature, then for this mechanism, all the following shall be specified:

1) the clear specification of the semantics and syntax of the component including its unique identifier;

2) the strategy of how this mechanism guarantees that all necessary parts of the signature are protected by this component; and

3) the strategy of how to handle signatures containing components defined in the present document.

EXAMPLE: The objects defined in IETF RFC 4998 [i.10], Annex A are examples of such alternative mechanisms but they only handle points 1) and 2).

Annex F (informative): Change history

Date	Version	Information about changes
September 2022	V.0.0.1	First draft to be circulated ONLY to ETSI ESI TB members.
April 2023	V0.0.2	Alternative draft: • All the validation data is placed in one attribute (valData). There may be several instances of this attribute. • All the references to validation data are placed in one attribute (refs).
May 2023	V0.0.3	Replacement of DigAlgVal of v0.02 by x5t specified in IETF RFC 9360. Resolution of one error in a reference that could not be found.
January 2024	V0.0.4	Reformulation of how to incorporate arcTst and compute the input to its message imprint computation in generation and validation. This reformulation is aligned with the formulation in XAdES and JAdES. Reformulation of requirements on the content of valData: the former formulation could be interpreted as if both certificates and revocation data must be present. The presence of certificates and/or revocation data will depend on the contents of the rest of CB-AdES signature. This new reformulation makes it clear that this qualifying property must contain certificates, or revocation data, or both of them. Reuse of x5t header parameter specified in IETF RFC 9360. Aligment of component for counter signatures to IETF RFC 9338. CDDL rules for assigning labels to be registered by IANA to identifiers for making the rest of the CDDL rules easier to read.
June 2024	V0.0.5	Refactored taking as basis draft ETSI TS 119 182-1 v1.1.13. The requirement level on the content of kid header parameter has been changed to should. The clause on counter signatures has been implemented. Added annex on IANA considerations.
January 2025	V0.0.6	Implemented reactions to IANA selected experts comments.
April 2025	V0.0.7	Implemented reactions to final IANA selected experts comments and dispositions for Nowina comments.
May 2025	V0.0.8	Annex C changed according to suggestions from IANA experts for accommodating the IANA registry fields. Suppression of wrong mentions to JWS instead COSE.
June 2025	V0.0.9	Implementation of resolutions to comments raised by ESI TO to v0.0.7.
January 2026	V0.0.10	Implementation of resolutions for technical and editorial comments raised to v0.0.9.
January 2026	V0.0.10	Suppression of references to file containing the CDDL schema as this file has not been generated yet.
January 2026	V0.0.11	Implementation of resolutions for technical and editorial comments raised to v0.0.10.

History

Version	Date	Status
V1.1.1	March 2026	Publication