

**ETSI EN 319 122-1** V1.2.1 (2021-10)



**Electronic Signatures and Infrastructures (ESI);  
CADES digital signatures;  
Part 1: Building blocks and CADES baseline signatures**

---

**Reference**

REN/ESI-0019122-1v121

---

**Keywords**ASN.1, CAdES, electronic signature, profile,  
security**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

---

**Important notice**

The present document can be downloaded from:

<http://www.etsi.org/standards-search>

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at [www.etsi.org/deliver](http://www.etsi.org/deliver).

Users of the present document should be aware that the document may be subject to revision or change of status.

Information on the current status of this and other ETSI documents is available at

<https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx>

If you find errors in the present document, please send your comment to one of the following services:

<https://portal.etsi.org/People/CommitteeSupportStaff.aspx>

---

**Notice of disclaimer & limitation of liability**

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or other professional standard and applicable regulations.

No recommendation as to products and services or vendors is made or should be implied.

In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

---

**Copyright Notification**

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.

The content of the PDF version shall not be modified without the written authorization of ETSI.

The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI 2021.  
All rights reserved.

# Contents

Intellectual Property Rights .....	6
Foreword.....	6
Modal verbs terminology.....	7
Introduction .....	7
1 Scope .....	8
2 References .....	8
2.1 Normative references .....	8
2.2 Informative references.....	9
3 Definition of terms, symbols and abbreviations.....	10
3.1 Terms.....	10
3.2 Symbols.....	11
3.3 Abbreviations .....	11
4 General syntax.....	12
4.1 General requirements .....	12
4.2 The data content type.....	12
4.3 The signed-data content type.....	12
4.4 The SignedData type.....	12
4.5 The EncapsulatedContentInfo type .....	12
4.6 The SignerInfo type .....	13
4.7 ASN.1 Encoding.....	13
4.7.1 DER .....	13
4.7.2 BER .....	13
4.8 Other standard data structures .....	13
4.8.1 Time-stamp token format.....	13
4.8.2 Additional types.....	13
4.9 Attributes.....	14
5 Attribute semantics and syntax.....	14
5.1 CMS defined basic signed attributes .....	14
5.1.1 The content-type attribute .....	14
5.1.2 The message-digest attribute .....	14
5.2 Basic attributes for CADES signatures .....	15
5.2.1 The signing-time attribute .....	15
5.2.2 Signing certificate reference attributes .....	15
5.2.2.1 General requirements .....	15
5.2.2.2 ESS signing-certificate attribute .....	15
5.2.2.3 ESS signing-certificate-v2 attribute .....	15
5.2.3 The commitment-type-indication attribute.....	16
5.2.4 Attributes for identifying the signed data type.....	17
5.2.4.1 The content-hints attribute .....	17
5.2.4.2 The mime-type attribute.....	17
5.2.5 The signer-location attribute .....	18
5.2.6 Incorporating attributes of the signer .....	18
5.2.6.1 The signer-attributes-v2 attribute .....	18
5.2.6.2 claimed-SAML-assertion .....	20
5.2.6.3 signed-SAML-assertion .....	20
5.2.7 The countersignature attribute.....	20
5.2.8 The content-time-stamp attribute.....	21
5.2.9 The signature-policy-identifier attribute and the SigPolicyQualifierInfo type.....	21
5.2.9.1 The signature-policy-identifier attribute.....	21
5.2.9.2 The SigPolicyQualifierInfo type .....	22
5.2.10 The signature-policy-store attribute .....	24
5.2.11 The content-reference attribute .....	25

5.2.12	The content-identifier attribute.....	25
5.2.13	The cms-algorithm-protection attribute .....	25
5.3	The signature-time-stamp attribute.....	26
5.4	Attributes for validation data values.....	26
5.4.1	Introduction.....	26
5.4.2	OCSP responses.....	26
5.4.2.1	OCSP response types .....	26
5.4.2.2	OCSP responses within RevocationInfoChoices .....	26
5.4.3	CRLs.....	27
5.5	Attributes for long term availability and integrity of validation material.....	27
5.5.1	Introduction.....	27
5.5.2	The ats-hash-index-v3 attribute .....	27
5.5.3	The archive-time-stamp-v3 attribute.....	29
6	CADES baseline signatures .....	31
6.1	Signature levels .....	31
6.2	General requirements .....	32
6.2.1	Algorithm requirements .....	32
6.2.2	Notation for requirements.....	32
6.3	Requirements on components and services .....	34
6.4	Legacy CADES baseline signatures.....	37
<b>Annex A (normative): Additional Attributes Specification.....</b>		<b>38</b>
A.1	Attributes for validation data.....	38
A.1.1	Certificates validation data .....	38
A.1.1.1	The complete-certificate-references attribute .....	38
A.1.1.2	The certificate-values attribute.....	39
A.1.2	Revocation validation data .....	39
A.1.2.1	The complete-revocation-references attribute.....	39
A.1.2.2	The revocation-values attribute .....	41
A.1.3	The attribute-certificate-references attribute .....	42
A.1.4	The attribute-revocation-references attribute .....	43
A.1.5	Time-stamps on references to validation data .....	44
A.1.5.1	The time-stamped-certs-crls-references attribute.....	44
A.1.5.2	The CADES-C-timestamp attribute .....	45
A.2	Deprecated attributes.....	45
A.2.1	Usage of deprecated attributes.....	45
A.2.2	The other-signing-certificate attribute .....	46
A.2.3	The signer-attributes attribute.....	46
A.2.4	The archive-time-stamp attribute .....	46
A.2.5	The long-term-validation attribute.....	46
A.2.6	The ats-hash-index attribute.....	46
<b>Annex B (normative): Alternative mechanisms for long term availability and integrity of validation data.....</b>		<b>47</b>
<b>Annex C: Void .....</b>		<b>48</b>
<b>Annex D (normative): Signature Format Definitions Using X.680 ASN.1 Syntax.....</b>		<b>49</b>
<b>Annex E (informative): Example Structured Contents and MIME .....</b>		<b>56</b>
E.1	Use of MIME to Encode Data.....	56
E.1.1	MIME Structure .....	56
E.1.2	Header Information .....	56
E.1.3	Content Encoding.....	57
E.1.4	Multi-Part Content.....	57
E.2	S/MIME.....	57
E.2.1	Using S/MIME .....	57
E.2.2	Using application/pkcs7-mime.....	58

E.2.3	Using multipart/signed and application/pkcs7-signature.....	58
E.3	Use of MIME in the signature.....	59
<b>Annex F (informative):</b>	<b>Change History .....</b>	<b>61</b>
History .....		62

# 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 Web server (<https://ipr.etsi.org/>).

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™** and **LTE™** 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 European Standard (EN) has been produced by ETSI Technical Committee Electronic Signatures and Infrastructures (ESI).

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

**Part 1: "Building blocks and CADES baseline signatures";**

Part 2: "Extended CADES signatures".

The present document partly contains an evolved specification of the ETSI TS 101 733 [1] and ETSI TS 103 173 [i.1].

National transposition dates	
Date of adoption of this EN:	18 October 2021
Date of latest announcement of this EN (doa):	31 January 2022
Date of latest publication of new National Standard or endorsement of this EN (dop/e):	31 July 2022
Date of withdrawal of any conflicting National Standard (dow):	31 July 2022

---

## 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 910/2014 [i.13].

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, GSM 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.2]). See ETSI TR 119 100 [i.4] for getting guidance on how to use the present document within the aforementioned framework.

---

# 1 Scope

The present document specifies CAAdES digital signatures. CAAdES signatures are built on CMS signatures [7], by incorporation of signed and unsigned attributes, which fulfil certain common requirements (such as the long term validity of digital signatures, for instance) in a number of use cases.

The present document specifies the ASN.1 definitions for the aforementioned attributes as well as their usage when incorporating them to CAAdES signatures.

The present document specifies formats for CAAdES baseline signatures, which 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.

The present document defines four levels of CAAdES baseline signatures addressing incremental requirements to maintain the validity of the signatures over the long term, in a way that a certain level always addresses all the requirements addressed at levels that are below it. Each level requires the presence of certain CAAdES attributes, suitably profiled for reducing the optionality as much as possible.

Procedures for creation, augmentation and validation of CAAdES digital signatures are out of scope and specified in ETSI EN 319 102-1 [i.5]. Guidance on creation, augmentation and validation of CAAdES digital signatures including the usage of the different properties defined in the present document is provided in ETSI TR 119 100 [i.4].

The present document aims at supporting digital signatures in different regulatory frameworks.

NOTE: Specifically, but not exclusively, CAAdES digital signatures specified in the present document aim at supporting electronic signatures, advanced electronic signatures, qualified electronic signatures, electronic seals, advanced electronic seals, and qualified electronic seals as per Regulation (EU) No 910/2014 [i.13].

---

## 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 at <https://docbox.etsi.org/Reference>.

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] ETSI TS 101 733 (V2.2.1): "Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAAdES)".
- [2] IETF RFC 2045 (1996): "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies".
- [3] IETF RFC 2634 (1999): "Enhanced Security Services for S/MIME".
- [4] IETF RFC 3161 (2001): "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)".
- [5] IETF RFC 5035 (2007): "Enhanced Security Services (ESS) Update: Adding CertID Algorithm Agility".

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

NOTE: Obsoletes IETF RFC 3280.

- [7] IETF RFC 5652 (2009): "Cryptographic Message Syntax (CMS)".

NOTE: Obsoletes IETF RFC 3852.

- [8] IETF RFC 5755 (2010): "An Internet Attribute Certificate Profile for Authorization".

NOTE: Obsoletes IETF RFC 3281.

- [9] IETF RFC 5816 (2010): "ESSCertIDv2 Update for RFC 3161".

- [10] IETF RFC 5911 (2010): "New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME".

- [11] IETF RFC 5912 (2010): "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)".

NOTE: Updated by IETF RFC 6268.

- [12] IETF RFC 6268 (2011): "Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)".

- [13] IETF RFC 5940 (2010): "Additional Cryptographic Message Syntax (CMS) Revocation Information Choices".

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

NOTE: Obsoletes IETF RFC 2560.

- [15] Recommendation ITU-T X.520 (11/2008)/ISO/IEC 9594-6:2008: "Information technology - Open Systems Interconnection - The Directory: Selected attribute types".

- [16] Recommendation ITU-T X.680 (2008): "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation".

- [17] Recommendation ITU-T X.690 (2008): "Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)".

- [18] OASIS Standard: "Security Assertion Markup Language (SAML) V2.0".

- [19] IETF RFC 6211 (2011): "Cryptographic Message Syntax (CMS) Algorithm Identifier Protection Attribute".

## 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 are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

- [i.1] ETSI TS 103 173 (V2.2.1): "Electronic Signatures and Infrastructures (ESI); CADES Baseline Profile".

- [i.2] ETSI TR 119 000: "Electronic Signatures and Infrastructures (ESI); The framework for standardization of signatures: overview".
- [i.3] ETSI TR 119 001: "Electronic Signatures and Infrastructures (ESI); The framework for standardization of signatures; Definitions and abbreviations".
- [i.4] ETSI TR 119 100: "Electronic Signatures and Infrastructures (ESI); Guidance on the use of standards for signature creation and validation".
- [i.5] ETSI EN 319 102-1: "Electronic Signatures and Infrastructures (ESI); Procedures for Creation and Validation of AdES Digital Signatures; Part 1: Creation and Validation".
- [i.6] ETSI EN 319 122-2: "Electronic Signatures and Infrastructures (ESI); CAAdES digital signatures; Part 2: Extended CAAdES signatures".
- [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] ETSI TS 119 312: "Electronic Signatures and Infrastructures (ESI); Cryptographic Suites".
- [i.9] ETSI EN 319 422: "Electronic Signatures and Infrastructures (ESI); Time-stamping protocol and time-stamp token profiles".
- [i.10] ETSI TS 101 533-1: "Electronic Signatures and Infrastructures (ESI); Data Preservation Systems Security; Part 1: Requirements for Implementation and Management".
- [i.11] ETSI TS 119 612: "Electronic Signatures and Infrastructures (ESI); Trusted Lists".
- [i.12] Commission Decision 2009/767/EC of 16 October 2009 amended by CD 2010/425/EU of 28 July 2010, setting out measures facilitating the use of procedures by electronic means through the "points of single contact" under Directive 2006/123/EC of the European Parliament and of the Council on services in the internal market.
- [i.13] Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC. OJ L 257, 28.8.2014, p. 73-114.
- [i.14] IETF RFC 3851 (2004): "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification".
- [i.15] IETF RFC 4998 (2007): "Evidence Record Syntax (ERS)".
- [i.16] Void.
- [i.17] Recommendation ITU-T X.501 (2008)/ISO/IEC 9594-1 (2008): "Information technology - Open Systems Interconnection - The Directory: Models".
- [i.18] Recommendation ITU-T X.509 (2008)/ISO/IEC 9594-8 (2008): "Information technology - Open Systems Interconnection - The Directory: Public-key and Attribute Certificate frameworks".
- [i.19] Recommendation ITU-T X.683 (2008): "Information technology - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications".

---

## 3 Definition of terms, symbols and abbreviations

### 3.1 Terms

For the purposes of the present document, the terms given in ETSI TR 119 001 [i.3] and the following apply:

**CAAdES signature:** digital signature that satisfies the requirements specified within ETSI EN 319 122 part 1 or part 2 [i.6]

**Certificate Revocation List (CRL):** signed list indicating a set of public key certificates that are no longer considered valid by the certificate issuer

**digital signature:** data appended to, or cryptographic transformation (see cryptography) of a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery e.g. by the recipient

**digital signature value:** result of the cryptographic transformation of a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery e.g. by the recipient

**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: In the case of IETF RFC 3161 [4] protocol, the electronic time-stamp is referring to the `timeStampToken` field within the `TimeStampResp` element (the TSA's response returned to the requesting client).

**Legacy CADES 101 733 signature:** digital signature generated according to ETSI TS 101 733 (V2.2.1) [1]

**Legacy CADES baseline signature:** digital signature generated according to ETSI TS 103 173 (V2.2.1) [i.1]

**Legacy CADES signature:** legacy CADES 101 733 signature or a legacy CADES baseline signature

**signature augmentation policy:** set of rules, applicable to one or more digital signatures, that defines the technical and procedural requirements for their augmentation, in order to meet a particular business need, and under which the digital signature(s) can be determined to be conformant

**signature creation policy:** set of rules, applicable to one or more digital signatures, that defines the technical and procedural requirements for their creation, in order to meet a particular business need, and under which the digital signature(s) can be determined to be conformant

**signature policy:** signature creation policy, signature augmentation policy, signature validation policy or any combination thereof, applicable to the same signature or set of signatures

**signature validation policy:** set of rules, applicable to one or more digital signatures, that defines the technical and procedural requirements for their validation, in order to meet a particular business need, and under which the digital signature(s) can be determined to be valid

**validation data:** data that is used to validate a digital signature

## 3.2 Symbols

Void.

## 3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI TR 119 001 [i.3] and the following apply:

ATSV2            `archive-time-stamp` attribute

NOTE: As defined in clause A.2.4.

ATSV3            `archive-time-stamp-v3` attribute

NOTE: As defined in clause 5.5.3.

## 4 General syntax

### 4.1 General requirements

CAdES signatures shall build on Cryptographic Message Syntax (CMS), as defined in IETF RFC 5652 [7], by incorporation of signed and unsigned attributes as defined in clause 5.1.

CAdES signatures shall comply with clauses 2, 3, 4 and 5 of IETF RFC 5652 [7].

The following clauses list the types that are used in the attributes described in clause 5.1.

### 4.2 The data content type

The data content type shall be as defined in CMS (IETF RFC 5652 [7], clause 4). It is used to refer to arbitrary octet strings.

NOTE: The data content type is identified by the object identifier `id-data OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs7(7) 1 }`.

### 4.3 The signed-data content type

The signed-data content type shall be as defined in CMS (IETF RFC 5652 [7], clause 5). It represents the content to sign and one or more signature values.

### 4.4 The SignedData type

The SignedData type shall be as defined in CMS (IETF RFC 5652 [7], clause 5.1). The CMSVersion shall be set as specified in clause 5.1 of IETF RFC 5652 [7].

`SignedData.xxx` refers to the element `xxx` within the SignedData type, like for example `SignedData.certificates`, or `SignedData.crls`. In the same way, if `xxx` is of type `XXX`, `SignedData.xxx.yyy` is used to refer to the element `yyy` of type `XXX`, like for example `SignedData.crls.crl` or `SignedData.crls.other`.

NOTE: Clause 5.1 of IETF RFC 5652 [7] requires that the CMS SignedData version be set to 3 if certificates from SignedData is present AND (any version 1 attribute certificates are present OR any SignerInfo structures are version 3 OR `eContentType` from `encapContentInfo` is other than `id-data`). Otherwise, the CMS SignedData version is required to be set to 1.

### 4.5 The EncapsulatedContentInfo type

The EncapsulatedContentInfo type shall be as defined in CMS (IETF RFC 5652 [7], clause 5.2).

For the purpose of long-term validation, either the `eContent` should be present, or the data that is signed should be archived in such a way as to preserve any data encoding.

NOTE 1: It is important that the OCTET STRING used to generate the signature remains the same every time either the verifier or an arbitrator validates the signature.

NOTE 2: The `eContent` is optional in CMS:

- When it is present, this allows the signed data to be encapsulated in the SignedData structure which then contains both the signed data and the signature. However, the signed data can only be accessed by a verifier able to decode the ASN.1 encoded SignedData structure.

- When it is missing, this allows the signed data to be sent or stored separately from the signature, and the `SignedData` structure only contains the signature. Under these circumstances, the data object that is signed needs to be stored and distributed in such a way as to preserve any data encoding.

## 4.6 The `SignerInfo` type

The `SignerInfo` type of the digital signature shall be as defined in CMS (IETF RFC 5652 [7], clause 5.3).

The per-signer information is represented in the type `SignerInfo`. In the case of multiple parallel signatures, there is one instance of this field for each signer.

The degenerate case where there are no signers shall not be used.

## 4.7 ASN.1 Encoding

### 4.7.1 DER

Distinguished Encoding Rules (DER) for ASN.1 types shall be as defined in Recommendation ITU-T X.690 [17].

### 4.7.2 BER

If Basic Encoding Rules (BER) are used for some ASN.1 types, it shall be as defined in Recommendation ITU-T X.690 [17].

## 4.8 Other standard data structures

### 4.8.1 Time-stamp token format

The `TimeStampToken` type shall be as defined in IETF RFC 3161 [4] and updated by IETF RFC 5816 [9].

NOTE: Time-stamp tokens are profiled in ETSI EN 319 422 [i.9].

### 4.8.2 Additional types

The `VisibleString`, `BMPString`, `IA5String`, `GeneralizedTime` and `UTCTime` types shall be as defined in Recommendation ITU-T X.680 [16].

The `DirectoryString` type shall be as defined in Recommendation ITU-T X.520 [15].

The `AttributeCertificate` type shall be as defined in IETF RFC 5755 [8] which is compatible with the definition in Recommendation ITU-T X.509 [i.18].

The `ResponderID`, `OCSPResponse` and `BasicOCSPResponse` types shall be as defined in IETF RFC 6960 [14].

The `Name`, `Certificate` and `AlgorithmIdentifier` types shall be as defined in IETF RFC 5280 [6].

The `Attribute` type shall be as defined in IETF RFC 5280 [6] which is compatible with the definition in Recommendation ITU-T X.501 [i.17].

The `CertificateList` type shall be as defined in IETF RFC 5280 [6] which is compatible with the X.509 v2 CRL syntax in Recommendation ITU-T X.509 [i.18].

The `RevocationInfoChoices` type shall be as defined in IETF RFC 5652 [7].

## 4.9 Attributes

Clause 5 provides details on attributes specified within CMS (IETF RFC 5652 [7]), ESS (IETF RFC 2634 [3] and IETF RFC 5035 [5]), and defines new attributes for building CADES signatures.

The clause distinguishes between two main types of attributes: signed attributes and unsigned attributes. The first ones are attributes that are covered by the digital signature value produced by the signer using his/her private key, which implies that the signer has processed these attributes before creating the signature. The unsigned attributes are added by the signer, by the verifier or by other parties after the production of the signature. They are not secured by the signature in the `SignerInfo` element (the one computed by the signer); however they can be actually covered by subsequent times-stamp attributes.

Signed and unsigned attributes are stored, respectively, in the `signedAttrs` and `unsignedAttrs` fields of `SignerInfo` (see clause 4.6).

---

# 5 Attribute semantics and syntax

## 5.1 CMS defined basic signed attributes

### 5.1.1 The `content-type` attribute

#### Semantics

The `content-type` attribute is a signed attribute.

The `content-type` attribute indicates the type of the signed content.

#### Syntax

The `content-type` attribute shall be as defined in CMS (IETF RFC 5652 [7], clause 11.1).

NOTE: As stated in IETF RFC 5652 [7], the content of `ContentType` (the value of the attribute `content-type`) is the same as the `eContentType` of the `EncapsulatedContentInfo` value being signed.

### 5.1.2 The `message-digest` attribute

#### Semantics

The `message-digest` attribute is a signed attribute.

The `message-digest` attribute specifies the message digest of the content being signed.

#### Syntax

The `message-digest` attribute shall be as defined in CMS (IETF RFC 5652 [7], clause 11.2).

The message digest calculation process shall be as defined in CMS (IETF RFC 5652 [7], clause 5.4).

## 5.2 Basic attributes for CAdES signatures

### 5.2.1 The `signing-time` attribute

#### Semantics

The `signing-time` attribute is a signed attribute.

The `signing-time` attribute shall specify the time at which the signer claims to having performed the signing process.

#### Syntax

The `signing-time` attribute shall be as defined in CMS (IETF RFC 5652 [7], clause 11.3).

### 5.2.2 Signing certificate reference attributes

#### 5.2.2.1 General requirements

##### Semantics

The attributes specified in clauses below shall contain one reference to the signing certificate.

The attributes specified in clauses below may contain references to some of or all the certificates within the signing certificate path, including one reference to the trust anchor when this is a certificate.

For each certificate, these attributes shall contain a digest value.

NOTE 1: For instance, the signature validation policy can mandate other certificates to be present which can include all the certificates up to the trust anchor.

NOTE 2: IETF RFC 2634 [3] and IETF RFC 5035 [5] state that the first certificate in the sequence is the certificate used to verify the signature and that other certificates in the sequence can be attribute certificates or other certificate types.

#### 5.2.2.2 ESS `signing-certificate` attribute

##### Semantics

The ESS `signing-certificate` attribute is a signed attribute.

The ESS `signing-certificate` attribute is a signing certificate attribute using the SHA-1 hash algorithm.

##### Syntax

The `signing-certificate` attribute shall be as defined in Enhanced Security Services (ESS), IETF RFC 2634 [3], clause 5.4, and further specified in the present document.

NOTE 1: The `certHash` from `ESSCertID` is computed using SHA-1 over the entire DER encoded certificate (IETF RFC 2634 [3]).

The `policies` field shall not be used.

NOTE 2: The information in the `IssuerSerial` element 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.

#### 5.2.2.3 ESS `signing-certificate-v2` attribute

##### Semantics

The ESS `signing-certificate-v2` attribute is a signed attribute.

The ESS `signing-certificate-v2` attribute is a signing certificate attribute using a hash algorithm different from SHA-1.

### Syntax

The `signing-certificate-v2` attribute shall be as defined in "ESS Update: Adding CertID Algorithm Agility", IETF RFC 5035 [5], clause 4 and further specified in the present document.

NOTE 1: The `certHash` from `ESSCertID` is computed over the entire DER encoded certificate (IETF RFC 5035 [5]).

The `policies` field shall not be used.

NOTE 2: The information in the `IssuerSerial` element 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.

## 5.2.3 The `commitment-type-indication` attribute

### Semantics

The `commitment-type-indication` attribute shall be a signed attribute that qualifies the signed data object.

The `commitment-type-indication` attribute shall indicate a commitment made by the signer when signing the data object.

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: The specification of commitment type identifiers is outside the scope of the present document. For a list of predefined commitment type identifiers, see the document on signature policies, ETSI TS 119 172-1 [i.7].

### Syntax

The `commitment-type-indication` attribute shall contain exactly one component of `AttributeValue` type.

The `commitment-type-indication` attribute value shall be an instance of `CommitmentTypeIndication` ASN.1 type.

The `commitment-type-indication` attribute shall be identified by the `id-aa-ets-commitmentType` OID.

The `commitmentTypeQualifier` field provides means to include additional qualifying information on the commitment made by the signer.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-commitmentType OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 16 }
```

```
CommitmentTypeIndication ::= SEQUENCE {
  commitmentTypeId      CommitmentTypeIdentifier,
  commitmentTypeQualifier SEQUENCE SIZE (1..MAX) OF CommitmentTypeQualifier OPTIONAL
}
```

```
CommitmentTypeIdentifier ::= OBJECT IDENTIFIER
```

```
CommitmentTypeQualifier ::= SEQUENCE {
  commitmentQualifierId COMMITMENT-QUALIFIER.&id,
  qualifier              COMMITMENT-QUALIFIER.&Qualifier OPTIONAL
}
```

```

COMMITMENT-QUALIFIER ::= CLASS {
  &id          OBJECT IDENTIFIER UNIQUE,
  &Qualifier   OPTIONAL }
WITH SYNTAX {
  COMMITMENT-QUALIFIER-ID  &id
  [COMMITMENT-TYPE        &Qualifier] }

```

## 5.2.4 Attributes for identifying the signed data type

### 5.2.4.1 The content-hints attribute

#### Semantics

The `content-hints` attribute shall be a signed attribute.

The `content-hints` attribute shall not be used within a countersignature.

The `content-hints` attribute shall provide information on the innermost signed content of a multi-layer message where one content is encapsulated in another.

#### Syntax

The `content-hints` attribute shall be as defined in ESS (IETF RFC 2634 [3], clauses 1.3.4 and 2.9).

The `contentDescription` may be used to complement a `contentType` defined elsewhere.

When used to indicate the precise format of the data to be presented to the user:

- the `contentType` shall indicate the type of the associated content. It is an object identifier assigned by an authority that defines the content type; and
- when the `contentType` is `id-data` (see clause 4.2) the `contentDescription` shall define the presentation format.

When the format of the content is defined by MIME types:

- the `contentType` shall be `id-data` (see clause 4.2);
- the `contentDescription` shall be used to indicate the encoding and the intended presentation application of the data, in accordance with the rules defined in IETF RFC 2045 [2]; see annex E for an example of structured contents and MIME.

### 5.2.4.2 The mime-type attribute

#### Semantics

The `mime-type` attribute shall be a signed attribute.

The `mime-type` attribute shall not be used within a countersignature.

The `mime-type` attribute shall indicate the mime-type of the signed data.

NOTE 1: This attribute is similar in spirit to the `contentDescription` field of the `content-hints` attribute, but can be used without a multi-layered document.

#### Syntax

The `mime-type` attribute shall contain exactly one component of `AttributeValue` type.

The `mime-type` attribute value shall be an instance of `MimeType` ASN.1 type.

The `mime-type` attribute shall be identified by the `id-aa-ets-mimeType` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-mimeType OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4) etsi(0)
  electronic-signature-standard (1733) attributes(2) 1 }
```

```
MimeType ::= UTF8String
```

The `MimeType` shall indicate the encoding and the intended presentation application of the signed data. The content of `MimeType` shall be in accordance with the rules defined in IETF RFC 2045 [2].

NOTE 2: See annex E for an example of structured contents and MIME.

## 5.2.5 The `signer-location` attribute

### Semantics

The `signer-location` attribute shall be a signed attribute.

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

### Syntax

The `signer-location` attribute shall contain exactly one component of `AttributeValue` type.

The `signer-location` attribute value shall be an instance of `SignerLocation` ASN.1 type.

The `signer-location` attribute shall be identified by the `id-aa-ets-signerLocation` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-signerLocation OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 17 }
```

```
SignerLocation ::= SEQUENCE { -- at least one of the following shall be present
  countryName    [0] DirectoryString OPTIONAL, -- As used to name a Country in X.520
  localityName   [1] DirectoryString OPTIONAL, -- As used to name a locality in X.520
  postalAddress  [2] PostalAddress OPTIONAL
}
```

```
PostalAddress ::= SEQUENCE SIZE(1..6) OF DirectoryString{maxSize}
  -- maxSize parametrization as specified in X.683
```

At least one of the fields `countryName`, `localityName` or `postalAddress` shall be present.

The content of `countryName` should be used to name a country a specified in Recommendation ITU-T X.520 [15], clause 6.3.1.

The content of `localityName` should be used to name a locality a specified in Recommendation ITU-T X.520 [15], clause 6.3.2.

## 5.2.6 Incorporating attributes of the signer

### 5.2.6.1 The `signer-attributes-v2` attribute

#### Semantics

The `signer-attributes-v2` attribute shall be a signed attribute.

The signer attributes shall encapsulate signer attributes (e.g. role). This attribute may encapsulate:

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

## Syntax

The `signer-attributes-v2` attribute shall contain exactly one component of `AttributeValue` type.

The `signer-attributes-v2` attribute value shall be an instance of `SignerAttributeV2` ASN.1 type.

The `signer-attributes-v2` attribute shall be identified by the `id-aa-ets-signerAttrV2` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-signerAttrV2 OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4)
  etsi(0) cadres(19122) attributes(1) 1 }
```

```
SignerAttributeV2 ::= SEQUENCE {
  claimedAttributes [0] ClaimedAttributes OPTIONAL,
  certifiedAttributesV2 [1] CertifiedAttributesV2 OPTIONAL,
  signedAssertions [2] SignedAssertions OPTIONAL
}
```

```
ClaimedAttributes ::= SEQUENCE OF Attribute
```

```
CertifiedAttributesV2 ::= SEQUENCE OF CHOICE {
  attributeCertificate [0] AttributeCertificate,
  otherAttributeCertificate [1] OtherAttributeCertificate
}
```

```
OtherAttributeCertificate ::= SEQUENCE {
  otherAttributeCertID OTHER-ATTRIBUTE-CERT.&id,
  otherAttributeCert OTHER-ATTRIBUTE-CERT.&OtherAttributeCert OPTIONAL
}
```

```
OTHER-ATTRIBUTE-CERT ::= CLASS {
  &id OBJECT IDENTIFIER UNIQUE,
  &OtherAttributeCert OPTIONAL }
WITH SYNTAX {
  OTHER-ATTRIBUTE-CERT-ID &id
  [OTHER-ATTRIBUTE-CERT-TYPE &OtherAttributeCert] }
```

```
SignedAssertions ::= SEQUENCE OF SignedAssertion
```

```
SignedAssertion ::= SEQUENCE {
  signedAssertionID SIGNED-ASSERTION.&id,
  signedAssertion SIGNED-ASSERTION.&Assertion OPTIONAL
}
```

```
SIGNED-ASSERTION ::= CLASS {
  &id OBJECT IDENTIFIER UNIQUE,
  &Assertion OPTIONAL }
WITH SYNTAX {
  SIGNED-ASSERTION-ID &id
  [SIGNED-ASSERTION-TYPE &Assertion] }
```

`Attribute` and `AttributeCertificate` shall be as defined in clause 4.8.2.

The `claimedAttributes` field shall contain a sequence of attributes claimed by the signer but which are not certified. These signer attributes are expressed using `Attribute` types.

NOTE 1: A user who wants to add a claimed role attribute can use the `RoleAttribute` as defined in Recommendation ITU-T X.509 [i.18].

NOTE 2: Clause 5.2.6.2 defines a new attribute that can be used to describe a claimed role by encapsulating a SAML assertion.

The `certifiedAttributes` field shall contain a non-empty sequence of certified attributes. These signer attributes shall be expressed by:

- `attributeCertificate`: an attribute certificate issued to the signer by an Attribute Authority (AA); or
- `otherAttributeCertificate`: an attribute certificate (issued, in consequence, by an Attribute Authority) in different syntax than the one used for X509 attribute certificates. The definition of specific `otherAttributeCertificates` is outside of the scope of the present document.

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

NOTE 3: 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 may be less restrictive than an attribute certificate.

An example of a definition of a specific `signedAssertions` is provided in clause 5.2.6.3. Any assertion encapsulated within this sequence shall be signed by third party.

NOTE 4: A possible content of such a qualifier can be a signed SAML assertion, see of SAML [18].

Empty `signer-attributes-v2` shall not be created.

### 5.2.6.2 claimed-SAML-assertion

#### Semantics

The `claimed-SAML-assertion` is a claimed assertion that shall include a SAML assertion.

The `claimed-SAML-assertion` may be used in a `claimedAttributes` field of the `signer-attributes-v2` attribute. It shall not be used anywhere else.

#### Syntax

The `claimed-SAML-assertion` is of ASN.1 type `Attribute`.

The `claimed-SAML-assertion` shall contain exactly one component of `AttributeValue` type.

The `claimed-SAML-assertion` value shall be an instance of `ClaimedSAMLAssertion` ASN.1 type.

The `claimed-SAML-assertion` shall be identified by the `id-aa-ets-claimedSAML` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-claimedSAML OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4)
    etsi(0) cades(19122) attributes(1) 2 }
```

```
ClaimedSAMLAssertion ::= OCTET STRING
```

The value of `ClaimedSAMLAssertion` shall contain the byte representation of SAML assertion as defined in SAML [18].

### 5.2.6.3 signed-SAML-assertion

The `signed-SAML-assertion` shall be identified by the `id-ets-signedSAML` OID.

The `signed-SAML-assertion` shall be of type `OCTET STRING`.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-ets-signedSAML OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4)
    etsi(0) cades(19122) additional(3) 0 }
```

```
SignedSAMLAssertion ::= OCTET STRING
```

The value of `ClaimedSAMLAssertion` shall contain the byte representation of a signed SAML assertion as defined in SAML [18].

## 5.2.7 The countersignature attribute

#### Semantics

The countersignature attribute is an unsigned attribute.

The countersignature attribute shall include a counter signature on the CADES signature where this attribute is included.

**Syntax**

The `countersignature` attribute shall be as defined in CMS (IETF RFC 5652 [7], clause 11.4).

**5.2.8 The content-time-stamp attribute****Semantics**

The `content-time-stamp` attribute shall be a signed attribute.

The `content-time-stamp` attribute shall encapsulate one time-stamp token of the signed data content before it is signed.

**Syntax**

The `content-time-stamp` attribute shall contain exactly one component of `AttributeValue` type.

The `content-time-stamp` attribute value shall be an instance of `ContentTimestamp` ASN.1 type.

The `content-time-stamp` attribute shall be identified by the `id-aa-ets-contentTimestamp` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-contentTimestamp OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 20 }
```

```
ContentTimestamp ::= TimeStampToken
```

The value of `messageImprint` of `TimeStampToken` (see clause 4.8.1) shall be a hash of:

- the value of `eContent` in the case of an attached signature; or
- the external data in the case of a detached signature.

In both cases, the hash shall be computed over the raw data, without ASN.1 tag and length.

**5.2.9 The signature-policy-identifier attribute and the SigPolicyQualifierInfo type****5.2.9.1 The signature-policy-identifier attribute****Semantics**

The `signature-policy-identifier` attribute shall be a signed attribute.

The `signature-policy-identifier` shall contain an explicit identifier of the signature policy.

**Syntax**

The `signature-policy-identifier` attribute shall contain exactly one component of `AttributeValue` type.

The `signature-policy-identifier` attribute value shall be an instance of `SignaturePolicyIdentifier` ASN.1 type.

The `signature-policy-identifier` attribute shall be identified by the `id-aa-ets-sigPolicyId` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-sigPolicyId OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
  rsadsi(113549) pkcs(1) pkcs9(9) smime(16) id-aa(2) 15 }
```

```
SignaturePolicyIdentifier ::= CHOICE {
  signaturePolicyId      SignaturePolicyId,
  signaturePolicyImplied SignaturePolicyImplied -- not used in this version
```

```

}
SignaturePolicyId ::= SEQUENCE {
    sigPolicyId      SigPolicyId,
    sigPolicyHash    SigPolicyHash,
    sigPolicyQualifiers SEQUENCE SIZE (1..MAX) OF SigPolicyQualifierInfo OPTIONAL
}
SignaturePolicyImplied ::= NULL
SigPolicyId ::= OBJECT IDENTIFIER
SigPolicyHash ::= OtherHashAlgAndValue
OtherHashAlgAndValue ::= SEQUENCE {
    hashAlgorithm AlgorithmIdentifier,
    hashValue      OtherHashValue }
OtherHashValue ::= OCTET STRING

```

The `signaturePolicyImplied` choice shall not be used.

The `sigPolicyId` field shall contain an object-identifier that uniquely identifies a specific version of the signature policy.

The `sigPolicyHash` field shall contain the identifier of the hash algorithm, and either the hash of the value of the signature policy or a zero-hash value.

A zero-hash value shall be an octet string of any length (including zero length) whose octets all have the value zero.

A zero-hash value shall be used to indicate that the policy hash value is not known.

If the `hashValue` field of the `sigPolicyHash` field contains a zero-hash value, signature validation applications shall interpret that value as indicating that the policy hash value is not known.

Signature creation applications that generate a zero-hash value should generate it with a length consistent with the hash algorithm specified by the `hashAlgorithm` field of the `sigPolicyHash` field.

NOTE 1: The use of a zero-hash value in the `hashValue` of the `sigPolicyHash` is to ensure backwards compatibility with earlier versions of ETSI TS 101 733 [1].

NOTE 2: Earlier versions of the present document were unclear on what exactly constitutes a zero-hash value, with the consequence that different implementations chose values of different length. The present document therefore requires that zero-hash values of any length have to be accepted. The recommendation to create zero-hash values with a length consistent with the specified hash algorithm is for compatibility with existing implementations - in particular those created prior to the introduction of zero-hash values - that may be unprepared to handle hash values with a different length.

NOTE 3: Depending on the hash algorithm, the actual computed hash value of a signature policy document may theoretically (although exceedingly unlikely) happen to be zero. Where applicable, applications can reject policy documents that would result in a zero-hash value, as the present document requires such values to be interpreted as an unknown hash value.

The input to hash computation of `sigPolicyHash` depends on the technical specification of the signature policy. In the case where the specification is not clear from the context of the signature, the `sp-doc-specification` qualifier shall be used to identify the used specification.

The `sigPolicyQualifiers` field may further qualify the `signature-policy-identifier` attribute. It contains a sequence of instances of `SigPolicyQualifierInfo` type which is defined in clause 5.2.9.2.

The `sigPolicyQualifiers` field may contain one or more qualifiers of the same type.

### 5.2.9.2 The SigPolicyQualifierInfo type

#### Semantics

The `SigPolicyQualifierInfo` type may be used to further qualify the `signature-policy-identifier` attribute.

Three qualifiers for the signature policy have been identified so far:

- a URI or URL where a copy of the signature policy document can be obtained (an element of type SPuri);
- a user notice that should be displayed whenever the signature is validated (an element of type SPUserNotice); and
- an identifier of the technical specification that defines the syntax used for producing the signature policy document (an element of type SPDocSpecification).

### Syntax

The ASN.1 definition of the SigPolicyQualifierInfo qualifier shall be as defined in annex D and is copied here for information:

```
SigPolicyQualifierInfo ::= SEQUENCE {
  sigPolicyQualifierId  SIG-POLICY-QUALIFIER.&id ({SupportedSigPolicyQualifiers}),
  qualifier             SIG-POLICY-QUALIFIER.&Qualifier
    ({SupportedSigPolicyQualifiers} {@sigPolicyQualifierId}) OPTIONAL
}

SupportedSigPolicyQualifiers SIG-POLICY-QUALIFIER ::= { noticeToUser |
  pointerToSigPolSpec | sigPolDocSpecification }

SIG-POLICY-QUALIFIER ::= CLASS {
  &id          OBJECT IDENTIFIER UNIQUE,
  &Qualifier   OPTIONAL }
WITH SYNTAX {
  SIG-POLICY-QUALIFIER-ID  &id
  [SIG-QUALIFIER-TYPE      &Qualifier] }

noticeToUser SIG-POLICY-QUALIFIER ::= {
  SIG-POLICY-QUALIFIER-ID id-spq-ets-notice SIG-QUALIFIER-TYPE SPUserNotice }

pointerToSigPolSpec SIG-POLICY-QUALIFIER ::= {
  SIG-POLICY-QUALIFIER-ID id-spq-ets-uri SIG-QUALIFIER-TYPE SPuri }

sigPolDocSpecification SIG-POLICY-QUALIFIER ::= {
  SIG-POLICY-QUALIFIER-ID id-spq-ets-docspec SIG-QUALIFIER-TYPE SPDocSpecification }
```

The semantics and syntax of the qualifier is as identified by the object-identifier in the sigPolicyQualifierId field. The ASN.1 definition of the qualifiers shall be as defined in annex D and is copied here for information:

```
-- spuri
id-spq-ets-uri OBJECT IDENTIFIER ::= { iso(1)
  member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
  smime(16) id-spq(5) 1 }

SPuri ::= IA5String

-- sp-user-notice
id-spq-ets-notice OBJECT IDENTIFIER ::= { iso(1)
  member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
  smime(16) id-spq(5) 2 }

SPUserNotice ::= SEQUENCE {
  noticeRef      NoticeReference OPTIONAL,
  explicitText   DisplayText OPTIONAL
}

NoticeReference ::= SEQUENCE {
  organization   DisplayText,
  noticeNumbers  SEQUENCE OF INTEGER
}

DisplayText ::= CHOICE {
  visibleString  VisibleString  (SIZE (1..200)),
  bmpString      BMPString      (SIZE (1..200)),
  utf8String     UTF8String     (SIZE (1..200))
}

-- sp-doc-specification
id-spq-ets-docspec OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4)
  etsi(0) cades(19122) id-spq (2) 1 }
```

```
SPDocSpecification ::= CHOICE {
  oid OBJECT IDENTIFIER,
  uri IA5String
}
```

An element of type `SPuri` 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.

An element of type `SPUserNotice` shall contain information that is intended for being displayed whenever the signature is validated.

The `explicitText` field shall contain the text of the notice to be displayed.

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

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

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

If the technical specification is identified using an OID, then the `oid` choice shall be used to contain the OID of the specification.

If the technical specification is identified using a URI, then the `uri` choice shall be used to contain this URI.

NOTE 3: This qualifier allows identifying whether the signature policy document is human readable, XML encoded, or ASN.1 encoded, by identifying the specific technical specifications where these formats will be defined.

## 5.2.10 The signature-policy-store attribute

### Semantics

The `signature-policy-store` attribute shall be an unsigned attribute.

The `signature-policy-store` attribute shall contain either:

- the signature policy document which is referenced in the `signature-policy-identifier` attribute 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

The `signature-policy-store` attribute shall contain exactly one component of `AttributeValue` type.

The `signature-policy-store` attribute value shall be an instance of `SignaturePolicyStore` ASN.1 type.

The `signature-policy-store` attribute shall be identified by the `id-aa-ets-sigPolicyStore` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-sigPolicyStore OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4)
  etsi(0) cadres(19122) attributes(1) 3 }
```

```
SignaturePolicyStore ::= SEQUENCE {
  spDocSpec SPDocSpecification,
  spDocument SignaturePolicyDocument
}
```

```
SignaturePolicyDocument ::= CHOICE {
  sigPolicyEncoded OCTET STRING,
  sigPolicyLocalURI IA5String
}
```

The `spDocument` shall contain the encoded signature policy document as content of the `sigPolicyEncoded` element, or an URI to a local store where the present document can be retrieved as `sigPolicyLocalURI`.

NOTE 1: Contrary to the `SPuri`, the `sigPolicyLocalURI` points to a local file.

The `spDocSpec` shall identify the technical specification that defines the syntax of the signature policy. The `SPDocSpecification` shall be as defined in clause 5.2.9.2.

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

NOTE 3: Being an unsigned attribute, the `signature-policy-store` attribute is not protected by the digital signature. If the `signature-policy-identifier` attribute is incorporated to the signature and contains in `sigPolicyHash` the digest value of the signature policy document, any alteration of the signature policy document present within `signature-policy-store` attribute or within a local store, would be detected by the failure of the digests comparison.

## 5.2.11 The content-reference attribute

### Semantics

The `content-reference` attribute is a signed attribute.

The `content-reference` attribute shall link one `SignedData` element to another.

### Syntax

The `content-reference` attribute shall be as defined in ESS (IETF RFC 2634 [3], clauses 1.3.4 and 2.11).

The `content-reference` attribute is a link from one `SignedData` to another. It is used to link a reply to the original message to which it refers, or to incorporate by reference one `SignedData` into another.

## 5.2.12 The content-identifier attribute

### Semantics

The `content-identifier` attribute is a signed attribute.

The `content-identifier` attribute provides an identifier for the signed content, for use when a reference may be later required to that content; for example, in the `content-reference` attribute in other signed data sent later.

### Syntax

The `content-identifier` attribute shall have attribute value `ContentIdentifier` as defined in ESS (IETF RFC 2634 [3], clauses 1.3.4 and 2.7).

The minimal `content-identifier` attribute should contain a concatenation of user-specific identification information (such as a user name or public keying material identification information), a `GeneralizedTime` string, and a random number.

## 5.2.13 The cms-algorithm-protection attribute

### Semantics

The `cms-algorithm-protection` attribute is a signed attribute.

The `cms-algorithm-protection` attribute shall contain and protect the digest algorithm and signature algorithm used.

### Syntax

The `cms-algorithm-protection` attribute shall be as defined in IETF RFC 6211 [19]).

## 5.3 The signature-time-stamp attribute

### Semantics

The `signature-time-stamp` attribute shall be an unsigned attribute.

The `signature-time-stamp` attribute shall encapsulate one time-stamp token computed on the digital signature value for a specific signer.

### Syntax

The `signature-time-stamp` attribute shall contain exactly one component of `AttributeValue` type.

The `signature-time-stamp` attribute value shall be an instance of `SignatureTimeStampToken` ASN.1 type.

The `signature-time-stamp` attribute shall be identified by the `id-aa-signatureTimeStampToken` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-signatureTimeStampToken OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 14 }
```

```
SignatureTimeStampToken ::= TimeStampToken
```

The value of the `messageImprint` field within the `TimeStampToken` shall be the hash value of the `signature` field (without the ASN.1 tag and length) within `SignerInfo` for which the `signature-time-stamp` attribute is created.

NOTE: In the case of multiple signatures, it is possible to have a:

- `signature-time-stamp` computed for each and all signers; or
- `signature-time-stamp` on some signers' signatures and none on other signers' signatures.

## 5.4 Attributes for validation data values

### 5.4.1 Introduction

The present document specifies different places where to incorporate missing validation data. See clauses 5.5 and A.1 for additional details.

For some types of validation data, the following clauses specify additional requirements when incorporating them into the signature.

### 5.4.2 OCSP responses

#### 5.4.2.1 OCSP response types

An OCSP response shall be incorporated into the signature either by using the encoding of the `OCSPResponse` type or the `BasicOCSPResponse` type as defined in clause 4.8.2.

The `OCSPResponse` type should be used.

#### 5.4.2.2 OCSP responses within `RevocationInfoChoices`

The `RevocationInfoChoices` type shall be as defined in clause 4.8.2.

OCSP responses shall be included within the `other` field of the `RevocationInfoChoices` type.

OCSP responses should be added using the encoding of `OCSPResponse` as specified in IETF RFC 5940 [13].

### 5.4.3 CRLs

Certificate Revocation Lists (CRLs) shall be as defined in IETF RFC 5280 [6].

## 5.5 Attributes for long term availability and integrity of validation material

### 5.5.1 Introduction

The present document specifies an archive-time-stamp attribute that uses the `ats-hash-index-v3` attribute. Both attributes are unsigned.

The archive-time-stamp attribute corresponds to a single `SignerInfo` element, including all its counter signatures. It protects the corresponding `SignerInfo` element, and all data from the `SignedData` needed to validate the `SignerInfo` element.

### 5.5.2 The `ats-hash-index-v3` attribute

#### Semantics

For the purpose of long term availability and integrity of validation data in the context of the present document, the `ats-hash-index-v3` attribute shall be an unsigned attribute of the CMS signature of the time-stamp token included in the `archive-time-stamp-v3` attribute as defined in clause 5.5.3.

The `ats-hash-index-v3` attribute shall provide an unambiguous imprint of the essential components of a CADES signature for use in the archive time-stamp.

When validating the `archive-time-stamp-v3`, first the contained `ats-hash-index-v3` shall be validated. All the hash values of all of the certificates, revocation information and unsigned attributes are recalculated. Only those which match one of the hash values in the instance of the `ATSHashIndexV3` type are known to be protected by the corresponding archive time-stamp. The validation of the `archive-time-stamp-v3` requires to have all the original values referenced in the `ats-hash-index-v3` attribute. The `ats-hash-index-v3` is invalid if it contains a reference for which the original value is not found, i.e.:

- a reference represented by an entry in `certificatesHashIndex` which corresponds to no instance of `CertificateChoices` within `certificates` field of the root `SignedData`;
- a reference represented by an entry in `crlsHashIndex` which corresponds to no instance of `RevocationInfoChoice` within `crls` field of the root `SignedData`; or
- a reference represented by an entry in `unsignedAttrValuesHashIndex` which corresponds to no octet stream resulting from concatenating one of the `AttributeValue` instances within field `Attribute.attrValues` and the corresponding `Attribute.attrType` within one `Attribute` instance in `unsignedAttrs` field of the `SignerInfo`.

Once the `ats-hash-index-v3` is validated, the `archive-time-stamp-v3` can be validated by recalculating the message imprint in the same way as in the creation of the attribute.

#### Syntax

The `ats-hash-index-v3` attribute shall contain exactly one component of `AttributeValue` type.

The `ats-hash-index-v3` attribute shall be DER encoded (see clause 4.7.1).

The `ats-hash-index-v3` attribute value shall be an instance of `ATSHashIndexV3` ASN.1 type.

The `ats-hash-index-v3` attribute shall be identified by the `id-aa-ATSHashIndex-v3` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ATSHashIndex-v3 OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4)
  etsi(0) cadres(19122) attributes(1) 5 }
```

```
ATSHashIndexV3 ::= SEQUENCE {
  hashIndAlgorithm      AlgorithmIdentifier,
  certificatesHashIndex SEQUENCE OF OCTET STRING,
  crlsHashIndex         SEQUENCE OF OCTET STRING,
  unsignedAttrValuesHashIndex SEQUENCE OF OCTET STRING
}
```

The elements covered by the `ats-hash-index-v3` attribute are included in the following ASN.1 SET OF structures: `unsignedAttrs`, `SignedData.certificates`, and `SignedData.crls`, where the `SignedData` field is the one of the CADES signature being archive time-stamped.

NOTE 1: The `SignedData.crls` component (see clause 4.4) can include OCSP and/or CRL revocation information.

The field `hashIndAlgorithm` shall contain an identifier of the hash algorithm used to compute the hash values contained in `certificatesHashIndex`, `crlsHashIndex`, and `unsignedAttrValuesHashIndex`. This algorithm shall be the same as the hash algorithm used for computing the message imprint included in the time-stamp token enveloped in the archive time-stamp unsigned attribute.

NOTE 2: ETSI TS 119 312 [i.8] provides guidance on the choice of hash algorithms.

The field `certificatesHashIndex` shall be a sequence of octet strings. Each one shall contain the hash value of one instance of `CertificateChoices` within the `certificates` field of the root `SignedData`. A hash value for every instance of `CertificateChoices`, as present at the time when the corresponding archive time-stamp is requested, shall be included in `certificatesHashIndex`. No other hash value shall be included in this field.

The field `crlsHashIndex` shall be a sequence of octet strings. Each one shall contain the hash value of one instance of `RevocationInfoChoice` within the `crls` field of the root `SignedData`. A hash value for every instance of `RevocationInfoChoice`, as present at the time when the corresponding archive time-stamp is requested, shall be included in `crlsHashIndex`. No other hash values shall be included in this field.

NOTE 3: The encoding of `certificatesHashIndex` and `crlsHashIndex` have the value empty and length zero, if the signature contains, respectively, no corresponding `CertificateChoices` or `RevocationInfoChoice` instance.

The field `unsignedAttrValuesHashIndex` shall be a sequence of octet strings. The sequence shall contain one octet string for every component within the `attrValues` field in every instance of `Attribute` contained in the `unsignedAttrs` field as present at time when the corresponding archive time-stamp is requested. No other octet string shall be included in this field. Each octet string shall contain the hash value of the octets resulting from concatenating the corresponding `Attribute.attrType` field and one of the instances of `AttributeValue` within the `Attribute.attrValues` field.

NOTE 4: The idea is that the `unsignedAttrValueHashIndex` covers all instances of `AttributeValue` of all instances of `Attribute` within the `unsignedAttrs` field separately so that there is no problem when in the future new attributes or new attribute values are added.

Each of the aforementioned hash values shall be the result of a hash computation on the entire component or the concatenation of the entire encoded components including their tag, length and value octets. Instances of `OtherCertificateFormat` shall be encoded in DER (see clause 4.7.1), whilst preserving the encoding of any signed field included in the `otherCert` item.

NOTE 5: Use of the `ats-hash-index-v3` attribute makes it possible to add additional certificate / revocation information / unsigned attribute or value within an unsigned attribute within `SignedData.certificates` / `SignedData.crls` / `unsignedAttrs` of the CADES signature (for instance counter signatures or further archive time-stamps), after an archive time-stamp has been applied to a signature, without invalidating such an archive time-stamp. Its use also allows the inclusion of components required by parallel signatures at a later time.

NOTE 6: In case a countersignature attribute is contained in a signature protected by an ATSV3, the adding of a new countersignature in the same attribute or as a new countersignature attribute is possible. However, the adding of a countersignature as an unsigned attribute to an existing countersignature that is protected by an ATSV3 will break the ATSV3 protection, because it changes the hash of the original countersignature attributed covered by the `ats-hash-index-v3` attribute.

NOTE 7: Figure 1 illustrates the computation of the `ats-hash-index-v3` and its combination with the ATSV3.

### 5.5.3 The `archive-time-stamp-v3` attribute

#### Semantics

The `archive-time-stamp-v3` attribute shall be an unsigned attribute.

The `archive-time-stamp-v3` attribute shall be a time-stamp token of the signed document and the signature, including signed attributes, and all other essential components of the signature as protected by the `ats-hash-index-v3` attribute.

#### Syntax

The `archive-time-stamp-v3` attribute shall contain exactly one component of `AttributeValue` type.

The `archive-time-stamp-v3` attribute value shall be an instance of `ArchiveTimeStampToken` ASN.1 type.

The `archive-time-stamp-v3` attribute shall be identified by the `id-aa-signatureTimeStampToken` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-archiveTimeStampV3 OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4)
  etsi(0) electronic-signature-standard(1733) attributes(2) 4 }
```

```
ArchiveTimeStampToken ::= TimeStampToken
```

The input for the `archive-time-stamp-v3`'s message imprint computation shall be the concatenation (in the order shown by the list below) of the signed data hash (see step 2 below) and certain fields in their binary encoded form without any modification and including the tag, length and value octets:

- 1) The `SignedData.encapContentInfo.eContentType`.
- 2) The octets representing the hash of the signed data. The hash is computed on the same content that was used for computing the hash value that is encapsulated within the `message-digest` signed attribute of the CAdES signature being archive-time-stamped. The hash algorithm applied shall be the same as the hash algorithm used for computing the archive time-stamp's message imprint. The hash algorithm identifier should be included in the `SignedData.digestAlgorithms` set.

NOTE 1: To validate the `archive-time-stamp-v3`, the hash of the signed data, as defined in point 2) is needed. In case of detached signatures, the hash can be provided from an external trusted source.

- 3) The fields `version`, `sid`, `digestAlgorithm`, `signedAttrs`, `signatureAlgorithm`, and `signature` within the `SignedData.signerInfos`'s item corresponding to the signature being archive time-stamped, in their order of appearance.
- 4) A single instance of `ATSHashIndexV3` type (as defined in clause 5.5.2) contained in the `ats-hash-index-v3` attribute.

The `archive-time-stamp-v3` shall include as an unsigned attribute a single `ats-hash-index-v3` attribute containing the instance included in step 4.

NOTE 2: The inclusion of the `ats-hash-index-v3` unsigned attribute's component in the process that builds the input to the computation of the archive time-stamp's message imprint ensures that all the essential components of the signature (including certificates, revocation information, and unsigned attributes) are protected by the time-stamp.

The items included in the hashing procedure and the concatenation order are shown in figure 1.

NOTE 3: When validated, an `archive-time-stamp-v3` unsigned attribute is a proof of existence at the time indicated in its time-stamp token, of the items that have contributed to the computation of its message imprint. This proof of existence can be used in validation procedures to ensure that signature validation is based on objects that truly existed in the past. This, for example, protects against a private signing key being compromised after the associated public key certificate expires resulting in the signature being considered invalid.

NOTE 4: Counter-signatures stored in `countersignature` attributes do not require independent archive time-stamps as they are protected by the archive time-stamp as an unsigned attribute.

Before incorporating a new `archive-time-stamp-v3` attribute, the `SignedData` (see clause 4.4) shall be extended to include any validation data, not already present, which is required for validating the signature being archive time-stamped. Validation data may include certificates, CRLs, OCSP responses, as required to validate any signed object within the signature including the existing signature, counter-signatures, time-stamps, OCSP responses, certificates, attribute certificates and signed assertions. In the case that the validation data contains a Delta CRL, then the whole set of CRLs shall be included to provide a complete revocation list.

NOTE 5: Validation data already present for example in the time-stamp token need not be included again.

The present document specifies two strategies for the inclusion of validation data, depending on whether attributes for long term availability, as defined in different versions of ETSI TS 101 733 [1], have already been added to the `SignedData`:

- If none of `ATSV2` attributes (see clause A.2.4), or an earlier form of archive time-stamp as defined in ETSI TS 101 733 [1] or `long-term-validation` (see clause A.2.5) attributes is already present in any `SignerInfo` of the root `SignedData`, then the new validation material shall be included within the root `SignedData.certificates`, or `SignedData.crls` as applicable.
- If an `ATSV2`, or other earlier form of archive time-stamp or a `long-term-validation` attribute, is present in any `SignerInfo` of the root `SignedData` then the root `SignedData.certificates` and `SignedData.crls` contents shall not be modified. The new validation material shall be provided within the `TimeStampToken` of the latest archive time-stamp (which can be an `ATSV2` as defined in ETSI TS 101 733 [1], or an `ATSV3`) or within the latest `long-term-validation` attribute (defined in ETSI TS 101 733 [1]) already contained in the `SignerInfo`, by one of the following methods:
  - the TSU provides the information in the `SignedData` of the timestamp token;
  - adding the `certificate-values` attribute and the `revocation-values` attribute as unsigned attributes within the `TimeStampToken`.

NOTE 6: In the case where an `ATSV2`, or other earlier form of archive time-stamp or a `long-term-validation` attribute, is present, once an `ATSV3` is added, "the latest archive time-stamp already contained in the `SignerInfo`" will be of type `ATSV3`.

If an `ATSV2`, or other earlier form of archive time-stamp or a `long-term-validation` attribute, is present then no other attributes than `ATSV3` or attributes specified as per annex B shall be added to the `unsignedAttrs`. During the validation, these `ATSV3` attributes or attributes specified as per annex B shall be first validated, and subsequently ignored for the validation of the older archive time-stamp or `long-term-validation` attributes.

OCSP responses shall be included as defined in clause 5.4.2.

If the OCSP response is included within `SignedData.crls`, it shall be included as defined in clause 5.4.2.2.

When generating a new attribute to include validation data, either initially when creating the signature or later when augmenting the signature, it shall be encoded in DER (see clause 4.7.1), whilst preserving the encoding of any signed field included in the attribute. The augmentation shall preserve the binary encoding of already present unsigned attributes and any component contributing to the archive time-stamp's message imprint computation input. When adding any new attribute after the signature was protected by an `ATSV3`, the new attributes should be DER encoded.

NOTE 7: In case the encoding of any of the elements protected by the `ats-hash-index-v3` attribute, is changed, the validation of the `ats-hash-index-v3` attribute will fail, because the corresponding hash value is not found. The encoding may change in case of BER encoded elements, which are reencoded.

Figure 1 illustrates the hashing process.

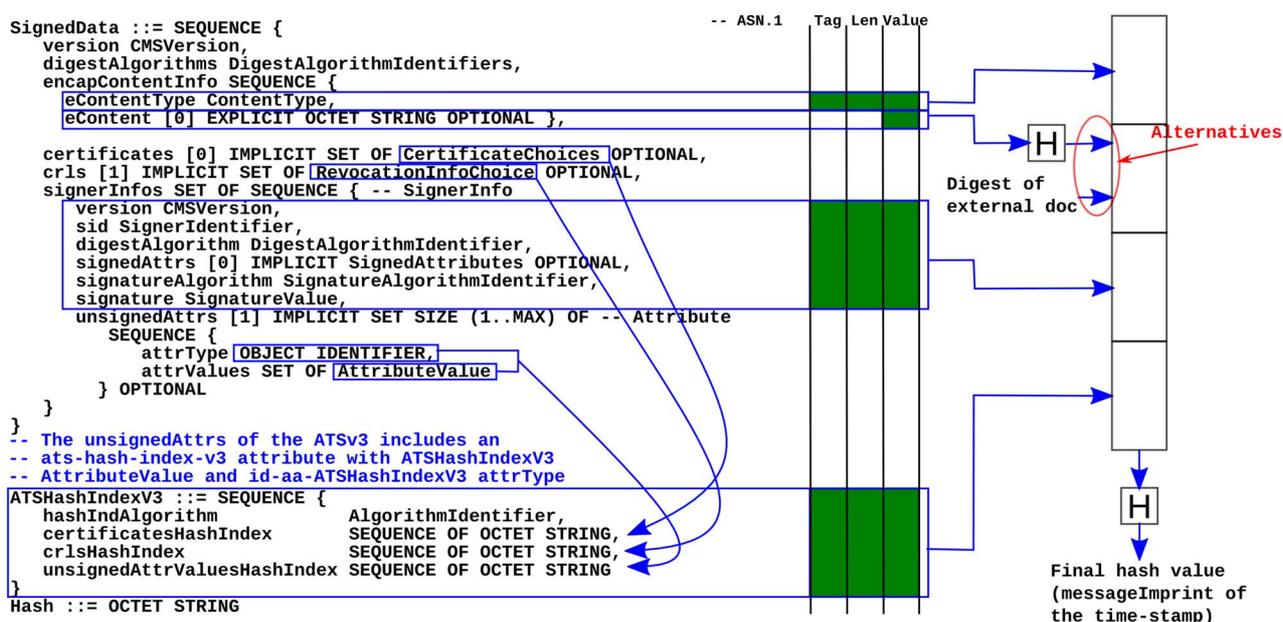


Figure 1: Hashing process

## 6 CAdES baseline signatures

### 6.1 Signature levels

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

- B-B level provides requirements for the incorporation of signed and some unsigned attributes when the signature is actually generated.
- B-T level provides requirement 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.
- 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.
- B-LTA level provides requirements for the incorporation of time-stamp tokens 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.4] 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. 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.10] or IETF RFC 4998 [i.15]) that can indicate other preservation techniques. Annex B defines what needs to be taken into account when using other techniques for long term availability and integrity of validation data and including a new unsigned attribute 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 [i.8].

NOTE: Cryptographic suites recommendations defined in ETSI TS 119 312 [i.8] 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 CADES signature levels.

The requirements on the attributes and certain signature fields for each CADES signature level are expressed in table 1. A row in the table either specifies requirements for an attribute, a signature field or a service.

A service can be provided by different attributes or other mechanisms (service provision options hereinafter). In this case, the specification of the requirements for a service is provided by two or more rows. The first row contains the requirements of the service. The requirements for the attributes and/or mechanisms used to provide the service are stated in the following rows.

Table 1 contains 8 columns. Below follows a detailed explanation of their meanings and contents:

- 1) Column "Attribute/Field/Service":
  - 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 attribute or signature field provides a service, this cell contains "SPO" (for Service Provision Option), followed by the name of the attribute or signature field.
  - c) Otherwise, this cell contains the name of the attribute or signature field.
- 2) Column "Presence in B-B-Level": This cell contains the specification of the presence of the attribute or signature field, or the provision of a service, for CADES-B-B signatures.
- 3) Column "Presence in B-T level": This cell contains the specification of the presence of the attribute or signature field, or the provision of a service, for CADES-B-T signatures.
- 4) Column "Presence in B-LT level": This cell contains the specification of the presence of the attribute or signature field, or the provision of a service, for CADES-B-LT signatures.
- 5) Column "Presence in B-LTA level": This cell contains the specification of the presence of the attribute or signature field, or the provision of a service, for CADES-B-LTA signatures.

Below follows 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 attribute or signature field shall be present, 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 attribute or signature field shall not be present. In these cases the content of the "Cardinality" column can indicate, the cardinality for each level if this value is not the same for all the levels. See example at the end of the present clause.
- "may be present": means that the attribute or signature field may be present, 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 attributes or signature fields.
- "conditioned presence": means that the presence of the item identified in the first column is conditioned as per the requirement(s) specified in column "Requirements" and requirements referenced by column "References" with the cardinality indicated in column "Cardinality".
- "\*": means that the attribute or the signature field (service) identified in the first column should not be present (provided) in the corresponding level. Upper signature levels may specify other requirements.

NOTE: Adding an unsigned attribute that is marked with a "\*" to a signature can lead to cases where a higher level cannot be achieved, except by removing the corresponding unsigned attribute.

- 6) Column "Cardinality": This cell indicates the cardinality of the attribute or the signature field. 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 follow the values indicating the cardinality:
- **0**: The signature shall not incorporate any instance of the attribute or signature field.
  - **1**: The signature shall incorporate exactly one instance of the attribute or signature field.
  - **0 or 1**: The signature shall incorporate zero or one instance of the attribute or signature field.
  - **≥ 0**: The signature shall incorporate zero or more instances of the attribute or signature field.
  - **≥ 1**: The signature shall incorporate one or more instances of the attribute or signature field.
- 7) Column "References": This cell contains either the number of the clause specifying the attribute in the present document, or a reference to the document and clause that specifies the signature field.
- 8) Column "Additional notes and requirements": This cell contains numbers referencing notes and/or letters referencing additional requirements on the attribute or the signature field. Both notes and additional requirements are listed below table 1.

EXAMPLE: In table 1, the row corresponding to `complete-certificate-references` attribute has a value "\*" in the cells in columns "Presence in B-B level" and "Presence in B-T level", and "shall not be present" in cells in columns "Presence in B-LT level" and "Presence in B-LTA level". The cell in column "Cardinality" indicates the cardinality for each level as follows: "B-B, B-T: 0 or 1" indicates that CADES-B-B and CADES-B-T signatures can incorporate one instance of `complete-certificate-references` attribute; "B-LT, B-LTA: 0" indicates that CADES-B-LT and CADES-B-LTA do not incorporate the `complete-certificate-references` attribute.

## 6.3 Requirements on components and services

Table 1 shows the presence and cardinality requirements on the attributes, signature fields, and services indicated in the first column for the four CADES baseline signature levels, namely: CADES-B-B, CADES-B-T, CADES-B-LT and CADES-B-LTA. Additional requirements are detailed below the table suitably labelled with the letter indicated in the last column.

NOTE 1: CADES-B-B signatures that incorporate only the elements/qualifying properties that are mandatory in table 1, and that implement the mandatory requirements, contain the lowest number of elements/qualifying properties, with the consequent benefits for interoperability.

**Table 1: Requirements for CADES-B-B, CADES-B-T, CADES-B-LT and CADES-B-LTA signatures**

Signature fields / Attributes / 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
SignedData.certificates	shall be present	shall be present	shall be present	shall be present	1	Clause 4.4	a, b, c, d, e 2, 3, 4
content-type	shall be present	shall be present	shall be present	shall be present	1	Clause 5.1.1	f
message-digest	shall be present	shall be present	shall be present	shall be present	1	Clause 5.1.2	
Service: protection of signing certificate	shall be provided	shall be provided	shall be provided	shall be provided	1	Clause 5.2.2	
SPO: ESS signing-certificate	conditioned presence	conditioned presence	conditioned presence	conditioned presence	0 or 1	Clause 5.2.2.2	g, h, j
SPO: ESS signing-certificate-v2	conditioned presence	conditioned presence	conditioned presence	conditioned presence	0 or 1	Clause 5.2.2.3	g, i, j
signing-time	shall be present	shall be present	shall be present	shall be present	1	Clause 5.2.1	
commitment-type-indication	may be present	may be present	may be present	may be present	0 or 1	Clause 5.2.3	
Service: identifying the signed data type	should be present	should be present	should be present	should be present	0 or 1	Clause 5.2.4	t, 6,7
SPO: content-hints	conditioned presence	conditioned presence	conditioned presence	conditioned presence	0 or 1	Clause 5.2.4.1	
SPO: mime-type	conditioned presence	conditioned presence	conditioned presence	conditioned presence	0 or 1	Clause 5.2.4.2	
signer-location	may be present	may be present	may be present	may be present	0 or 1	Clause 5.2.5	
signer-attributes-v2	may be present	may be present	may be present	may be present	0 or 1	Clause 5.2.6.1	
countersignature	may be present	may be present	may be present	may be present	≥ 0	Clause 5.2.7	
content-time-stamp	may be present	may be present	may be present	may be present	≥ 0	Clause 5.2.8	5
signature-policy-identifier	may be present	may be present	may be present	may be present	0 or 1	Clause 5.2.9.1	
signature-policy-store	conditioned presence	conditioned presence	conditioned presence	conditioned presence	0 or 1	Clause 5.2.10	k
content-reference	may be present	may be present	may be present	may be present	0 or 1	Clause 5.2.11	
content-identifier	may be present	may be present	may be present	may be present	0 or 1	Clause 5.2.12	
cms-algorithm-protection	may be present	may be present	may be present	may be present	0 or 1	Clause 5.2.13	8
signature-time-stamp	*	shall be present	shall be present	shall be present	≥ 1	Clause 5.3	l, m, 5

Signature fields / Attributes / 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
certificate-values	*	*	shall not be present	shall not be present	B-B, B-T: 0 or 1 B-LT, B-LTA: 0	Clause A.1.1.2	
complete-certificate-references	*	*	shall not be present	shall not be present	B-B, B-T: 0 or 1 B-LT, B-LTA: 0	Clause A.1.1.1	g
revocation-values	*	*	shall not be present	shall not be present	B-B, B-T : 0 or 1 B-LT, B-LTA: 0	Clause A.1.2.2	
complete-revocation-references	*	*	shall not be present	shall not be present	B-B, B-T: 0 or 1 B-LT, B-LTA: 0	Clause A.1.2.1	
attribute-certificate-references	*	*	shall not be present	shall not be present	B-B, B-T : 0 or 1 B=LT, B=LTA: 0	Clause A.1.3	j, n
attribute-revocation-references	*	*	shall not be present	shall not be present	B-B, B-T: 0 or 1 B-LT, B-LTA: 0	Clause A.1.4	n
CAeS-C-timestamp	*	*	shall not be present	shall not be present	B-B, B-T: ≥ 0 B-LT, B-LTA: 0	Clause A.1.5.2	5
time-stamped-certs-crls-references	*	*	shall not be present	shall not be present	B-B, B-T: ≥ 0 B-LT, B-LTA: 0	Clause A.1.5.1	5
Service: revocation values in long-term validation	*	*	shall be provided	shall be provided	1		o, p
SPO: SignedData.crls.crl	*	*	conditioned presence	conditioned presence	0 or 1	Clause 4.4	q
SPO: SignedData.crls.other	*	*	conditioned presence	conditioned presence	0 or 1	Clause 4.4	r
archive-time-stamp-v3	*	*	*	shall be provided	≥ 1	Clause 5.5.3	s

## Additional requirements:

- a) Requirement for `SignedData.certificates`. The generator shall include the signing certificate in the `SignedData.certificates` field.
- b) Requirement for `SignedData.certificates`. In order to facilitate path building, the generator should include in the `SignedData.certificates` field all certificates not available to verifiers that can be used during path building.
- c) Requirement for `SignedData.certificates`. In the case that the signature is meant to be validated through a Trusted List as specified in ETSI TS 119 612 [i.11] the generator should include all intermediary certificates forming a chain between the signing certificate and a CA present in the Trusted List, which are not available to verifiers.
- d) Requirement for `SignedData.certificates`. The generator shall include the full set of certificates, including the trust anchors when they are available in the form of certificates that have been used to validate the signature. This set includes certificates required for validating the signing certificate, for validating any attribute certificate present in the signature, for validating revocation information (i.e. OCSP response and CRL) if certificates are not already included, and for validating any time-stamp token's signing certificate (i.e. a TSA certificate) already incorporated to the signature.
- e) Requirement for `SignedData.certificates`. Duplication of certificate values within the signature should be avoided.
- f) Requirement for `content-type`. The `content-type` attribute shall have value `id-data` (see clause 4.2).
- g) Requirement for `SPO: ESS signing-certificate`, `SPO: ESS signing-certificate-v2`, and `complete-certificate-references`. The `issuerSerial` field should not be included in the encoding of the `ESSCertID`, `ESSCertIDv2` or `OtherCertID` type.
- h) Requirement for `SPO: ESS signing-certificate`. The `ESS signing-certificate` attribute shall be used if the SHA-1 hash algorithm is used.
- i) Requirement for `SPO: ESS signing-certificate-v2`. The `ESS signing-certificate-v2` attribute shall be used when another hash algorithm than SHA-1 is used.
- j) Requirement for `SPO: ESS signing-certificate` and `SPO: ESS signing-certificate-v2` and `attribute-certificate-references`. The generator should migrate to the use of `ESS signing-certificate-v2` in preference to `ESS signing-certificate` in line with the guidance regarding limited lifetime for the use of SHA-1 given in ETSI TS 119 312 [i.8].
- k) Requirement for `signature-policy-store`. The `signature-policy-store` attribute may be incorporated in the CAdES signature only if the `signature-policy-identifier` attribute is also incorporated and it contains in `sigPolicyHash` the digest value of the signature policy document, Otherwise the `signature-policy-store` shall not be incorporated in the CAdES signature.
- l) Requirement for `signature-time-stamp`. The generator shall use DER encoding (clause 4.7.1) for any `signature-time-stamp` attribute, whilst preserving the encoding of any other attribute field.
- m) Requirement for `signature-time-stamp`. The time-stamp tokens encapsulated within the `signature-time-stamp` attributes shall be created before the signing certificate has been revoked or has expired.
- n) Requirements for `attribute-certificate-references` and `attribute-revocation-references`. The `attribute-certificate-references` and `attribute-revocation-references` attributes may be used when a at least a certified signer attribute (`certifiedAttributesV2` as defined in clause 5.2.6.1) or a signed assertion (`signedAssertions` as defined in clause 5.2.6.1) is present within the signer attributes in the digital signature. Otherwise, `attribute-certificate-references` and `attribute-revocation-references` attributes shall not be used.

- o) Requirement for Service: revocation values in long-term validation. The generator shall include the full set of revocation data (CRL or OCSP responses) that have been used in the validation of the signature. This set includes all certificate status information required for validating the signing certificate, for validating any attribute certificate or signed assertion present in the signature, for validating revocation information (i.e. OCSP response and CRL) if they are not already included and for validating any time-stamp token's signing certificate (i.e. a TSA certificate) already incorporated to the signature.
- p) Requirement for Service: revocation values in long-term validation. Duplication of revocation values within the signature should be avoided.
- q) Requirement for SPO: `SignedData.crls.crl`. When the full set of revocation data contains CRLs [6], then the CRL values shall be included within `SignedData.crls.crl`.
- r) Requirement for SPO: `SignedData.crls.other`. When the full set of revocation data contains OCSP responses [14], then the OCSP response values shall be included within `SignedData.crls.other` as specified in IETF RFC 5940 [13].
- s) Requirement for `archive-time-stamp-v3`. Before generating and incorporating an `archive-time-stamp-v3` attribute, all the validation material required for verifying the signature, which are not already in the signature, shall be included. This validation material includes validation material used to validate previous archive time stamp.
- t) Requirement for Service: identifying the signed data type. At least one of the attributes, `content-hints` or `mime-type` should be present and shall describe the signed data type.

NOTE 2: On `SignedData.certificates`. A certificate is considered available to the verifier, if reliable information about its location is known and allows automated retrieval of the certificate (for instance through an Authority Info Access Extension or equivalent information present in a TSL).

NOTE 3: On `SignedData.certificates`. Requirement c) applies specifically but not exclusively to signing certificates that are EU qualified and supported by Trusted Lists as defined in CD 2009/767/EC [i.12] amended by CD 2010/425/EU.

NOTE 4: On `SignedData.certificates`. In the general case, different verifiers can have different trust parameters and can validate the signing certificate through different chains. Therefore, generators may not know which certificates will be relevant for path building. However, in practice, such certificates can often clearly be identified. In this case, it is advised that generators include them unless they can be automatically retrieved by verifiers.

NOTE 5: On `content-time-stamp`, `signature-time-stamp`, `CAAdES-C-timestamp`, and `time-stamped-certs-crls-references`. Several instances of this attribute can be incorporated to the signature, coming from different TSUs.

NOTE 6: Without the `mime-type`, the signed data might be interpreted in different ways. This might lead to misunderstandings when the data is shown in one way to the signer, and might be shown after the signature in a different way. Adding the `mime-type` used to show the document at the moment of signature can help avoiding such situations.

NOTE 7: In case of a detached signature, where the creator of the signature has no knowledge of the content of the signed data, the `mime-type application/octet-stream` can be used.

NOTE 8: This algorithm is useful, in case there exists two hash algorithms with the same output length, one being strong, the other being weak. In case the hash algorithm used in the signature is not signed, it might be stated later on, that the weak one was used.

## 6.4 Legacy CAAdES baseline signatures

When new unsigned attributes are incorporated to legacy CAAdES baseline signatures, these attributes shall comply with the present document.

## Annex A (normative): Additional Attributes Specification

### A.1 Attributes for validation data

#### A.1.1 Certificates validation data

##### A.1.1.1 The complete-certificate-references attribute

###### Semantics

The `complete-certificate-references` attribute shall be an unsigned attribute.

The `complete-certificate-references` attribute:

- 1) Shall contain the reference to the certificate of the trust anchor if such certificate does exist, and the references to CA certificates within the signing certificate path.
- 2) Shall not contain the reference to the signing certificate.

NOTE 1: The signer's certificate is referenced in the signing certificate attribute (see clause 5.2.2). May contain references to the certificates used to sign CRLs or OCSP responses for certificates referenced by references in 1), and references to certificates within their respective certificate paths.

- 3) Shall not contain references to CA certificates that pertain exclusively to the certificate paths of certificates used to sign attribute certificates or signed assertions within the `signer-attributes-v2` attribute.

NOTE 2: The references to certificates exclusively used in the validation of attribute certificate or signed assertions are stored in the `attribute-certificate-references` attribute (see clause A.1.3).

###### Syntax

The `complete-certificate-references` attribute shall contain exactly one component of `AttributeValue` type.

The `complete-certificate-references` attribute value shall be an instance of `CompleteCertificateRefs` ASN.1 type.

The `complete-certificate-references` attribute shall be identified by the `id-aa-ets-certificateRefs` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-certificateRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 21 }
```

```
CompleteCertificateRefs ::= SEQUENCE OF OtherCertID
```

```
OtherCertID ::= SEQUENCE {
  otherCertHash OtherHash,
  issuerSerial IssuerSerial OPTIONAL
}
```

```
OtherHash ::= CHOICE {
  sha1Hash OtherHashValue, -- This contains a SHA-1 hash
  otherHash OtherHashAlgAndValue
}
```

NOTE 3: Copies of the certificate values can be held using the `certificate-values` attribute, defined in clause A.1.1.2 or within `SignedData.certificates`.

This attribute may include references to the certification chain for any TSU that provides time-stamp tokens. In this case, the unsigned attribute shall be added to the `SignedData` of the relevant time-stamp token.

NOTE 4: In the case of a `content-time-stamp`, the time-stamp token cannot be changed after the signature without invalidating the signature. Consequently, this unsigned attribute needs to be added before signing.

### A.1.1.2 The `certificate-values` attribute

#### Semantics

The `certificate-values` attribute shall be an unsigned attribute.

The `certificate-values` attribute:

- 1) Shall contain the values of the certificates referenced within `complete-certificate-references` (clause A.1.1.1), `attribute-certificate-references` (clause A.1.3), and the `signing-certificate-reference` (clause 5.2.2) attributes, which are not stored `SignedData.certificates`. Certificate values within `SignedData.certificates` should not be included.
- 2) No other certificates shall be included.

#### Syntax

The `certificate-values` attribute shall contain exactly one component of `AttributeValue` type.

The `certificate-values` attribute value shall be an instance of `CertificateValues` ASN.1 type.

The `certificate-values` attribute shall be identified by the `id-aa-ets-certValues` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-certValues OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 23 }
```

```
CertificateValues ::= SEQUENCE OF Certificate
```

NOTE 1: `Certificate` is defined in IETF RFC 5280 [6] (see annex D) and is a basic syntax to include Recommendation ITU-T X.509 v3 [i.18] certificates.

This attribute may include the certification information for any TSUs that have provided the time-stamp tokens, if these certificates are not already included in the TSTs as part of the TSUs signatures. In this case, the unsigned attribute shall be added to the `SignedData` of the relevant time-stamp token.

NOTE 2: In the case of a `content-time-stamp`, the time-stamp token cannot be changed after the signature without invalidating the signature. Consequently, this unsigned attribute needs to be added before signing or somewhere else within the signature, if needed.

## A.1.2 Revocation validation data

### A.1.2.1 The `complete-revocation-references` attribute

#### Semantics

The `complete-revocation-references` attribute shall be an unsigned attribute.

The `complete-revocation-references` attribute:

- 1) Shall contain a reference to a revocation value for the signing certificate.
- 2) Shall contain the references to the revocation values (e.g. CRLs or OCSP values) corresponding to CA certificates references in the `complete-certificate-references` attribute, except for the trust anchors. It shall not contain references to revocation values for the trust anchor.

NOTE 1: A trust anchor is by definition trusted, thus no revocation information for the trust anchor is used during the validation.

- 3) May contain references to the revocation values corresponding to certificates used to sign CRLs or OCSP responses referenced in references from 1) and 2), and to certificates within their respective certificate paths.
- 4) Shall not contain references to the revocation values corresponding to CA certificates that pertain exclusively to the certificate paths of certificates used to sign attribute certificates or signed assertions within the `signer-attributes-v2` attribute.

NOTE 2: The references to revocation values exclusively used in the validation of attribute certificate or signed assertions are stored in the `attribute-revocation-references` attribute (see clause A.1.4).

The `complete-revocation-references` attribute should be used in preference to the `OtherRevocationInfoFormat` specified in IETF RFC 5652 [7] to maintain backwards compatibility with the earlier versions of ETSI TS 101 733 [1].

### Syntax

The `complete-revocation-references` attribute shall contain exactly one component of `AttributeValue` type.

The `complete-revocation-references` attribute value shall be an instance of `CompleteRevocationRefs` ASN.1 type.

The `complete-revocation-references` attribute shall be identified by the `id-aa-ets-revocationRefs` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-revocationRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 22 }
```

```
CompleteRevocationRefs ::= SEQUENCE OF CrlOcspRef
```

```
CrlOcspRef ::= SEQUENCE {
  crlids [0] CRLListID OPTIONAL,
  ocspids [1] OcspListID OPTIONAL,
  otherRev [2] OtherRevRefs OPTIONAL
}
```

```
CRLListID ::= SEQUENCE {
  crls SEQUENCE OF CrlValidatedID
}
```

```
CrlValidatedID ::= SEQUENCE {
  crlHash OtherHash,
  crlIdentifier CrlIdentifier OPTIONAL
}
```

```
CrlIdentifier ::= SEQUENCE {
  crlIssuer Name,
  crlIssuedTime UTCTime,
  crlNumber INTEGER OPTIONAL
}
```

```
OcspListID ::= SEQUENCE {
  ocspResponses SEQUENCE OF OcspResponsesID
}
```

```
OcspResponsesID ::= SEQUENCE {
  ocspIdentifier OcspIdentifier,
  ocspRefHash OtherHash OPTIONAL
}
```

```
OcspIdentifier ::= SEQUENCE {
  ocspResponderID ResponderID, -- As in OCSP response data
  producedAt GeneralizedTime -- As in OCSP response data
}
```

```
OtherRevRefs ::= SEQUENCE {
  otherRevRefType OTHER-REVOCATION-REF.&id,
  otherRevRefs SEQUENCE OF OTHER-REVOCATION-REF.&Type
}
```

```

}
OTHER-REVOCATION-REF ::= CLASS {
  &Type,
  &id OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
  WITH SYNTAX &Type ID &id }

```

In `CompleteRevocationRefs`, the sequence shall start with the `CrLOcspRef` for the signing certificate (see point 1) above. Subsequently, the `CrLOcspRef` entries corresponding to the values of point 2) above shall be added, in the same order as they appeared in the `complete-certificate-references` attribute. In the end, the `CrLOcspRef` elements corresponding to point 3) above may be added.

When creating a `crlValidatedID`, the `crlHash` shall be computed over the entire DER encoded CRL including the signature.

The `crlIdentifier` should be present unless the CRL can be inferred from other information.

The `crlIdentifier` shall identify the CRL using the issuer name and the CRL issued time, which shall correspond to the time `thisUpdate` contained in the issued CRL, and if present, the `crlNumber`.

In the case that the identified CRL is a Delta CRL, then references to the set of CRLs to provide a complete revocation list shall be included.

The `OcspIdentifier` shall identify the OCSPP response using the issuer name and the time of issue of the OCSPP response, which shall correspond to the time produced as contained in the issued OCSPP response.

The `ocspRefHash` should be included.

NOTE 3: In earlier versions of ETSI TS 101 733 [1], the `ocspRefHash` field was optional. In order to provide backward compatibility, the ASN.1 structure was not changed.

NOTE 4: The absence of the `ocspRefHash` field makes OCSPP responses substitutions attacks possible, if for instance OCSPP responder keys are compromised. In this case, out-of-band mechanisms can be used to ensure that none of the OCSPP responder keys have been compromised at the time of validation.

The `ocspRefHash` shall include the digest of the OCSPP responses using the types stated in clause 5.4.2.1.

NOTE 5: Copies of the CRL and OCSPP responses values can be held using the `revocation-values` attribute defined in clause A.1.2.2 or within `SignedData.crls`.

The syntax and semantics of other revocation references are outside the scope of the present document. The definition of the syntax of the other form of revocation information shall be as identified by `OtherRevRefType`.

This attribute may include the references to the full set of the CRL, or OCSPP responses that have been used to verify the certification chain for any TSUs that provide time-stamp tokens. In this case, the unsigned attribute shall be added to the `SignedData` of the relevant time-stamp token.

NOTE 6: In the case of a `content-time-stamp`, the time-stamp token cannot be changed after the signature without invalidating the signature. Consequently, this unsigned attribute needs to be added before signing.

## A.1.2.2 The `revocation-values` attribute

### Semantics

The `revocation-values` attribute shall be an unsigned attribute.

The `revocation-values` attribute:

- 1) Shall contain the elements corresponding to the references in `complete-revocation-references` (clause A.1.2.1) and `attribute-revocation-references` (clause A.1.4), which are not stored in `SignedData.crls`.
- 2) No other element shall be included.

The `revocation-values` attribute should be used in preference to the `OtherRevocationInfoFormat` specified in IETF RFC 5652 [7] to maintain backwards compatibility with the earlier version of ETSI TS 101 733 [1].

### Syntax

The `revocation-values` attribute shall contain exactly one component of `AttributeValue` type.

The `revocation-values` attribute value shall be an instance of `RevocationValues` ASN.1 type.

The `revocation-values` attribute shall be identified by the `id-aa-ets-revocationValues` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-revocationValues OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 24 }
```

```
RevocationValues ::= SEQUENCE {
  crlVals      [0] SEQUENCE OF CertificateList OPTIONAL,
  ocspsVals    [1] SEQUENCE OF BasicOCSPResponse OPTIONAL,
  otherRevVals [2] OtherRevVals OPTIONAL
}
```

```
OtherRevVals ::= SEQUENCE {
  otherRevValType OTHER-REVOCATION-VAL.&id,
  otherRevVals    SEQUENCE OF OTHER-REVOCATION-REF.&Type
}
```

```
OTHER-REVOCATION-VAL ::= CLASS {
  &Type,
  &id  OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
  WITH SYNTAX &Type ID &id }
```

The syntax and semantics of the contents of `OtherRevVals` field are outside the scope of the present document. The definition of the syntax of the other form of revocation information shall be as identified by `OtherRevRefType`.

`CertificateList` shall be as defined in clause 4.8.2.

OCSP responses shall be included using the types stated in clause 5.4.2.1.

If an OCSP response is of type `OCSPResponse`, it shall be included within `otherRevVals` using the OID `id-ri-ocsp-response` (1.3.6.1.5.5.7.16.2).

This attribute may include the values of revocation data including CRLs and OCSPs for any TSUs that have provided the time-stamp tokens, if these certificates are not already included in the TSTs as part of the TSUs signatures. In this case, the unsigned attribute shall be added to the `SignedData` of the relevant time-stamp token.

**NOTE:** In the case of a `content-time-stamp`, the time-stamp token cannot be changed after the signature without invalidating the signature. Consequently, this unsigned attribute needs to be added before signing or somewhere else within the signature, if needed.

## A.1.3 The attribute-certificate-references attribute

### Semantics

The `attribute-certificate-references` attribute shall be an unsigned attribute.

The `attribute-certificate-references` attribute:

- 1) Shall contain, if they are not present within `complete-certificate-references` attribute, the references to the trust anchor and the references to CA certificates within the path of the signing certificate(s) of the attribute certificate(s) and signed assertion(s) incorporated to the CADES signature. References present within `complete-certificate-references` attribute should not be included.

- 2) Shall contain, if they are not present within `complete-certificate-references` attribute, the reference(s) to the signing certificate(s) of the attribute certificate(s) and signed assertion(s) incorporated to the CAdES signature. References present within `complete-certificate-references` attribute should not be included.
- 3) May contain references to the certificates used to sign CRLs or OCSP responses and certificates within their respective certificate paths, which are used for validating the signing certificate(s) of the attribute certificate(s) and signed assertion(s) incorporated to the CAdES signature. References present within `complete-certificate-references` attribute should not be included.

### Syntax

The `attribute-certificate-references` attribute shall contain exactly one component of `AttributeValue` type.

The `attribute-certificate-references` attribute value shall be an instance of `AttributeCertificateRefs` ASN.1 type.

The `attribute-certificate-references` attribute shall be identified by the `id-aa-ets-attrCertificateRefs` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-attrCertificateRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 44 }
```

```
AttributeCertificateRefs ::= SEQUENCE OF OtherCertID
```

NOTE: Copies of the certificate values referenced here can be held using the `certificate-values` attribute defined in clause A.1.1.2 or within `SignedData.certificates`. The attribute certificate itself is stored in the `signer-attributes-v2` as defined in clause 5.2.6.1.

## A.1.4 The attribute-revocation-references attribute

### Semantics

The `attribute-revocation-references` attribute shall be an unsigned attribute.

The `attribute-revocation-references` attribute:

- 1) Shall contain, if they are not present within the `complete-revocation-references` attribute, the references to the revocation values corresponding to CA certificates within the path(s) of the signing certificate(s) of the attribute certificate(s) and signed assertion(s) incorporated to the CAdES signature. It shall not contain references to revocation values for the trust anchor. References present within `complete-revocation-references` attribute should not be included.

NOTE 1: A trust anchor is by definition trusted, thus no revocation information for the trust anchor is used during the validation.

- 2) Shall contain, if they are not present within the `complete-revocation-references` attribute, the references to the revocation value(s) for the signing certificate(s) of the attribute certificate(s) and signed assertion(s) incorporated to the CAdES signature. References present within `complete-revocation-references` attribute should not be included.
- 3) May contain references to the revocation values on certificates used to sign CRLs or OCSP responses and certificates within their respective certificate paths, which are used for validating the signing certificate(s) of the attribute certificate(s) and signed assertion(s) incorporated to the CAdES signature. References present within `complete-revocation-references` attribute should not be included.

## Syntax

The `attribute-revocation-references` attribute shall contain exactly one `AttributeValue`.

The `attribute-revocation-references` attribute value shall be an instance of `AttributeRevocationRefs` ASN.1 type.

The `attribute-revocation-references` attribute shall be identified by the `id-aa-ets-attrRevocationRefs` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-attrRevocationRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 45 }
```

```
AttributeRevocationRefs ::= SEQUENCE OF CrlOcspRef
```

NOTE 2: Copies of the CRL and OCSP responses values referenced here can be held using the `revocation-values` attribute defined in clause A.1.2.2 or within `SignedData.crls`.

Should one or more of the identified CRLs be a Delta CRL, this attribute shall include references to the set of CRLs required to provide complete revocation lists.

## A.1.5 Time-stamps on references to validation data

### A.1.5.1 The time-stamped-certs-crls-references attribute

#### Semantics

The `time-stamped-certs-crls-references` attribute shall be an unsigned attribute.

The `time-stamped-certs-crls-references` attribute shall encapsulate one time-stamp token of the `complete-certificate-references` attribute and the `complete-revocation-references` attribute.

#### Syntax

The `time-stamped-certs-crls-references` attribute shall contain exactly one component of `AttributeValue` type.

The `time-stamped-certs-crls-references` attribute value shall be an instance of `TimestampedCertsCRLs` ASN.1 type.

The `time-stamped-certs-crls-references` attribute shall be identified by the `id-aa-ets-certCRLTimestamp` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-certCRLTimestamp OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 26 }
```

```
TimestampedCertsCRLs ::= TimeStampToken
```

This attribute shall encapsulate one time-stamp token, whose `messageImprint` field shall be the hash of the concatenated values of the following data objects, as present within the electronic signature:

- `complete-certificate-references` attribute; and
- `complete-revocation-references` attribute.

Each attribute shall be included in the hash with the `attrType` and `attrValues` (including type and length) but without the type and length of the outer `SEQUENCE`.

The attributes being time-stamped should be encoded in DER (see clause 4.7.1). If DER is not employed, then the binary encoding of the ASN.1 structures being time-stamped should be preserved to ensure that the recalculation of the data hash is consistent.

For further information and definition of `TimeStampToken`, see clause 4.8.1.

## A.1.5.2 The `CADES-C-time-stamp` attribute

### Semantics

The `CADES-C-time-stamp` attribute shall be an unsigned attribute.

The `CADES-C-time-stamp` attribute shall encapsulate one time-stamp token covering the signature, the signature timestamp, the `complete-certificate-references` attribute; and `complete-revocation-references` attribute.

NOTE: This time-stamp covers the `CADES-E-C` level signature as defined in ETSI EN 319 122-2 [i.6].

### Syntax

The `CADES-C-time-stamp` attribute shall contain exactly one component of `AttributeValue` type.

The `CADES-C-time-stamp` attribute value shall be an instance of `ESCTimeStampToken` ASN.1 type.

The `CADES-C-time-stamp` attribute shall be identified by the `id-aa-ets-escTimeStamp` OID.

The corresponding definitions shall be as defined in annex D and are copied here for information.

```
id-aa-ets-escTimeStamp OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 25 }
```

```
ESCTimeStampToken ::= TimeStampToken
```

This attribute encapsulates one time-stamp token, whose `messageImprint` field shall be the hash of the concatenated values (without the ASN.1 type or length encoding for that value) of the following data objects:

- OCTETSTRING of the signature field within `SignerInfo`;
- `signature-time-stamp`;
- `complete-certificate-references` attribute; and
- `complete-revocation-references` attribute.

Each attribute shall be included in the hash with the `attrType` and `attrValues` (including type and length) but without the type and length of the outer SEQUENCE.

The attributes being time-stamped should be encoded in DER (see clause 4.7.1). If DER is not employed, then the binary encoding of the ASN.1 structures being time-stamped should be preserved to ensure that the recalculation of the data hash is consistent.

For further information and definition of `TimeStampToken`, see clause 4.8.1.

## A.2 Deprecated attributes

### A.2.1 Usage of deprecated attributes

Clause A.2 lists deprecated attributes. They are kept in the document to facilitate the handling of legacy signatures but they shall not be added any more to a signature. The only exception is the `long-term-validation` attribute that may still be added to signatures already containing a `long-term-validation` attribute.

## A.2.2 The other-signing-certificate attribute

The `other-signing-certificate` attribute as defined in ETSI TS 101 733 (V2.2.1) [1], is deprecated. Instead, the `signing-certificate-v2` attribute as defined in clause 5.2.2.3 shall be used.

## A.2.3 The signer-attributes attribute

The `signer-attributes` attribute as defined in ETSI TS 101 733 (V2.2.1) [1], is deprecated. Instead the `signer-attributes-v2` as defined in clause 5.2.6.1 shall be used.

## A.2.4 The archive-time-stamp attribute

The `archive-time-stamp` (ATSV2) attribute as defined in ETSI TS 101 733 (V2.2.1) [1], is deprecated. New ATSV2 attributes shall not be created. Systems may extend the lifetime of signatures containing ATSV2 attributes by incorporating new ATSV3 as described in clause 5.5.3.

## A.2.5 The long-term-validation attribute

The use of the `long-term-validation` attribute as defined in ETSI TS 101 733 (V2.2.1) [1], is deprecated. New `long-term-validation` attributes shall not be created. Systems may extend the lifetime of signatures containing `long-term-validation` attributes by incorporating new ATSV3 as defined in clause 5.5.3.

## A.2.6 The ats-hash-index attribute

The `ats-hash-index` attribute as defined in ETSI TS 101 733 (V2.2.1) [1], is deprecated. Instead the `ats-hash-index-v3` as defined in clause 5.5.2 shall be used.

**NOTE:** The ASN.1 definition of the `ats-hash-index` can lead to ambiguities in the decoding if the default hash algorithm is used and was not able to handle the case where values were added to unsigned attributes already covered by an ATSV3.

---

## Annex B (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 5.5.

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

- 1) The clear specification of the semantics and syntax of the attribute including its OID.
- 2) The strategy of how this mechanism guarantees that all necessary parts of the signature are protected by this attribute.
- 3) The strategy of how to handle signatures containing attributes defined in the present document. In particular, in case ATSV3 attributes are already included in the signature it shall be ensured that the previous time-stamp tokens within these attributes are not invalidated and that all validation material needed to validate the signature before the incorporation of the new attribute is incorporated into the signature and protected by the new attribute.
- 4) The strategy of how to handle legacy CADES signatures. In particular it shall be guaranteed that in case of previously added attributes for long term availability and integrity of validation data they are not invalidated.

NOTE 1: Such mechanisms, defined outside of the present document, can be used to provide long term availability and integrity of validation data. However, they do not represent CADES-B-LTA level as defined in clause 6 or CADES-E-A levels as defined in ETSI EN 319 122-2 [i.6].

NOTE 2: Such mechanisms might be included in future versions of the present document and assigned to a corresponding CADES level.

EXAMPLE: The attributes defined in IETF RFC 4998 [i.15], annex A are examples of such alternative mechanisms but they only handle points 1) and 2).

---

Annex C:  
Void

## Annex D (normative): Signature Format Definitions Using X.680 ASN.1 Syntax

In case of discrepancy in the ASN.1 definitions between the previous clauses and this annex, this annex takes precedence.

The following ASN.1 modules shall be interpreted using the syntax defined in Recommendation ITU-T X.680 [16].

The ASN.1 modules defined in this clause shall import the types and structures from IETF RFC 6268 [12], IETF RFC 5911 [10], IETF RFC 5912 [11], IETF RFC 6960 [14] and IETF RFC 3161 [4] as written in the import part of the module.

```

ETSI-CAdES-ExplicitSyntax97 { itu-t(0) identified-organization(4) etsi(0) cades(19122)
  id-mod(0) cades-explicit97(1)}

DEFINITIONS EXPLICIT TAGS ::=
BEGIN
-- EXPORTS All -

IMPORTS

-- Imports from Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the
-- Public Key Infrastructure Using X.509 (PKIX): IETF RFC 6268
-- (update for module from Imports from Cryptographic Message Syntax (CMS): IETF RFC 5652)
  ContentInfo, ContentType, id-data, id-signedData, SignedData, EncapsulatedContentInfo,
  SignerInfo, id-contentType, id-messageDigest, MessageDigest, id-signingTime, SigningTime,
  id-countersignature, Countersignature, RevocationInfoChoices, Attribute
  FROM CryptographicMessageSyntax-2010
  { iso(1) member-body(2) us(840) rsadsi(113549)
    pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) }

-- Imports from New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME:
-- IETF RFC 5911
-- (updated for module from Enhanced Security Services (ESS) Update: Adding CertID Algorithm Agility
-- IETF RFC 5035)
  id-aa-signingCertificate, SigningCertificate, IssuerSerial, id-aa-contentReference,
  ContentReference, id-aa-contentIdentifier, ContentIdentifier, id-aa-signingCertificateV2,
  SigningCertificateV2
  FROM ExtendedSecurityServices-2009
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) modules(0) id-mod-ess-2006-02(42) }

-- Imports from New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX):
-- IETF RFC 5912
-- (updated for module from Internet X.509 Public Key Infrastructure - Certificate and CRL
-- Profile: IETF RFC 5280)
  Certificate, AlgorithmIdentifier, CertificateList, Name
  FROM PKIX1Explicit-2009
  { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5)
    pkix(7) id-mod(0) id-mod-pkix1-explicit-02(51)}

  GeneralNames, GeneralName, PolicyInformation
  FROM PKIX1Implicit-2009
  { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5)
    pkix(7) id-mod(0) id-mod-pkix1-implicit-02(59)}

-- Imports from New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX):
-- IETF RFC 5912
-- (updated for module from Internet Attribute Certificate Profile for Authorization: IETF RFC 5755)
  AttributeCertificate
  FROM PKIXAttributeCertificate-2009
  { iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-attribute-cert-02(47)}

-- Imports from X.509 Internet Public Key Infrastructure - Online Certificate Status Protocol - OCSP
-- IETF RFC 6960
  BasicOCSPResponse, ResponderID
  FROM OCSP-2013-08
  { iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-ocsp-2013-08(82) }

-- Imports from Internet X.509 Public Key Infrastructure - Time-Stamp Protocol (TSP), IETF RFC 3161
  TimeStampToken

```

```

FROM PKIXTSP
  { iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-tsp(13)}
-- Imports from Information technology - Open Systems Interconnection -
-- The Directory: Selected attribute types - X.520
  DirectoryString{}
  FROM SelectedAttributeTypes
    { joint-iso-itu-t ds(5) module(1) selectedAttributeTypes(5) 6 }
;

-- Definitions of Object Identifier arcs used in the present document
-- =====

-- Object Identifier arc for attributes first defined in TS 101 733
id-etsi-es-attributes OBJECT IDENTIFIER ::=
  { itu-t(0) identified-organization(4) etsi(0)
    electronic-signature-standard (1733) attributes(2) }

-- Object Identifier arc for attributes first defined in the present document
id-etsi-cades-attributes OBJECT IDENTIFIER ::=
  { itu-t(0) identified-organization(4) etsi(0) cades(19122) attributes(1) }

-- Object Identifier arc for signature policy qualifier first defined in the present document
id-etsi-cades-spq OBJECT IDENTIFIER ::=
  { itu-t(0) identified-organization(4) etsi(0) cades(19122) id-spq(2) }

-- Object Identifier arc for ASN.1 modules defined in the present document
id-etsi-cades-mod OBJECT IDENTIFIER ::=
  { itu-t(0) identified-organization(4) etsi(0) cades(19122) id-mod(0) }

-- Attributes for basic CAdES signatures
-- =====

-- commitment-type attribute (clause 5.2.3)

id-aa-ets-commitmentType OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 16}

CommitmentTypeIndication ::= SEQUENCE {
  commitmentTypeId      CommitmentTypeIdentifier,
  commitmentTypeQualifier SEQUENCE SIZE (1..MAX) OF CommitmentTypeQualifier OPTIONAL
}

CommitmentTypeIdentifier ::= OBJECT IDENTIFIER

CommitmentTypeQualifier ::= SEQUENCE {
  commitmentQualifierId  COMMITMENT-QUALIFIER.&id,
  qualifier              COMMITMENT-QUALIFIER.&Qualifier OPTIONAL
}

COMMITMENT-QUALIFIER ::= CLASS {
  &id      OBJECT IDENTIFIER UNIQUE,
  &Qualifier OPTIONAL }
WITH SYNTAX {
  COMMITMENT-QUALIFIER-ID &id
  [COMMITMENT-TYPE      &Qualifier] }

-- mime-type attribute (clause 5.2.4.2)

id-aa-ets-mimeType OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4) etsi(0)
  electronic-signature-standard (1733) attributes(2) 1 }

MimeType ::= UTF8String

-- signer-location attribute (clause 5.2.5)

id-aa-ets-signerLocation OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 17 }

SignerLocation ::= SEQUENCE { -- at least one of the following shall be present
  countryName  [0] DirectoryString OPTIONAL, -- As used to name a Country in X.520
  localityName [1] DirectoryString OPTIONAL, -- As used to name a locality in X.520
  postalAddress [2] PostalAddress OPTIONAL
}

```

```

PostalAddress ::= SEQUENCE SIZE(1..6) OF DirectoryString{maxSize}
-- maxSize parametrization as specified in X.683

-- signer-attributes-v2 attribute (clause 5.2.6.1)

id-aa-ets-signerAttrV2 OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4)
etsi(0) cades(19122) attributes(1) 1 }

SignerAttributeV2 ::= SEQUENCE {
  claimedAttributes [0] ClaimedAttributes OPTIONAL,
  certifiedAttributesV2 [1] CertifiedAttributesV2 OPTIONAL,
  signedAssertions [2] SignedAssertions OPTIONAL
}

ClaimedAttributes ::= SEQUENCE OF Attribute

CertifiedAttributesV2 ::= SEQUENCE OF CHOICE {
  attributeCertificate [0] AttributeCertificate,
  otherAttributeCertificate [1] OtherAttributeCertificate
}

OtherAttributeCertificate ::= SEQUENCE {
  otherAttributeCertID OTHER-ATTRIBUTE-CERT.&id,
  otherAttributeCert OTHER-ATTRIBUTE-CERT.&OtherAttributeCert OPTIONAL
}

OTHER-ATTRIBUTE-CERT ::= CLASS {
  &id OBJECT IDENTIFIER UNIQUE,
  &OtherAttributeCert OPTIONAL }
WITH SYNTAX {
  OTHER-ATTRIBUTE-CERT-ID &id
  [OTHER-ATTRIBUTE-CERT-TYPE &OtherAttributeCert] }

SignedAssertions ::= SEQUENCE OF SignedAssertion

SignedAssertion ::= SEQUENCE {
  signedAssertionID SIGNED-ASSERTION.&id,
  signedAssertion SIGNED-ASSERTION.&Assertion OPTIONAL
}

SIGNED-ASSERTION ::= CLASS {
  &id OBJECT IDENTIFIER UNIQUE,
  &Assertion OPTIONAL }
WITH SYNTAX {
  SIGNED-ASSERTION-ID &id
  [SIGNED-ASSERTION-TYPE &Assertion] }

-- claimed-SAML-assertion attribute (clause 5.2.6.2)

id-aa-ets-claimedSAML OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4)
etsi(0) cades(19122) attributes(1) 2 }

ClaimedSAMLAssertion ::= OCTET STRING

-- content-timestamp attribute (clause 5.2.8)

id-aa-ets-contentTimestamp OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 20 }

ContentTimestamp ::= TimeStampToken

-- signature-policy-identifier attribute (clause 5.2.9.1)

id-aa-ets-sigPolicyId OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
rsadsi(113549) pkcs(1) pkcs9(9) smime(16) id-aa(2) 15 }

SignaturePolicyIdentifier ::= CHOICE {
  signaturePolicyId SignaturePolicyId,
  signaturePolicyImplied SignaturePolicyImplied -- not used in this version
}

SignaturePolicyId ::= SEQUENCE {
  sigPolicyId SigPolicyId,
  sigPolicyHash SigPolicyHash,
  sigPolicyQualifiers SEQUENCE SIZE (1..MAX) OF SigPolicyQualifierInfo OPTIONAL
}

```

```

}

SignaturePolicyImplied ::= NULL

SigPolicyId ::= OBJECT IDENTIFIER

SigPolicyHash ::= OtherHashAlgAndValue

OtherHashAlgAndValue ::= SEQUENCE {
    hashAlgorithm AlgorithmIdentifier,
    hashValue      OtherHashValue }

OtherHashValue ::= OCTET STRING

SigPolicyQualifierInfo ::= SEQUENCE {
    sigPolicyQualifierId  SIG-POLICY-QUALIFIER.&id ({SupportedSigPolicyQualifiers}),
    qualifier             SIG-POLICY-QUALIFIER.&Qualifier
    ({SupportedSigPolicyQualifiers} {@sigPolicyQualifierId}) OPTIONAL
}

SupportedSigPolicyQualifiers SIG-POLICY-QUALIFIER ::= { noticeToUser |
    pointerToSigPolSpec | sigPolDocSpecification }

SIG-POLICY-QUALIFIER ::= CLASS {
    &id          OBJECT IDENTIFIER UNIQUE,
    &Qualifier   OPTIONAL }
WITH SYNTAX {
    SIG-POLICY-QUALIFIER-ID &id
    [SIG-QUALIFIER-TYPE    &Qualifier] }

noticeToUser SIG-POLICY-QUALIFIER ::= {
    SIG-POLICY-QUALIFIER-ID id-spq-ets-notice SIG-QUALIFIER-TYPE SPUserNotice }

pointerToSigPolSpec SIG-POLICY-QUALIFIER ::= {
    SIG-POLICY-QUALIFIER-ID id-spq-ets-uri SIG-QUALIFIER-TYPE SPuri }

sigPolDocSpecification SIG-POLICY-QUALIFIER ::= {
    SIG-POLICY-QUALIFIER-ID id-spq-ets-docspec SIG-QUALIFIER-TYPE SPDocSpecification }

-- Signature policy qualifiers types (clause 5.2.9.2)
-- spuri
id-spq-ets-uri OBJECT IDENTIFIER ::= { iso(1)
    member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
    smime(16) id-spq(5) 1 }

SPuri ::= IA5String

-- sp-user-notice
id-spq-ets-notice OBJECT IDENTIFIER ::= { iso(1)
    member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
    smime(16) id-spq(5) 2 }

SPUserNotice ::= SEQUENCE {
    noticeRef      NoticeReference OPTIONAL,
    explicitText   DisplayText OPTIONAL
}

NoticeReference ::= SEQUENCE {
    organization   DisplayText,
    noticeNumbers  SEQUENCE OF INTEGER
}

DisplayText ::= CHOICE {
    visibleString  VisibleString (SIZE (1..200)),
    bmpString      BMPString      (SIZE (1..200)),
    utf8String     UTF8String     (SIZE (1..200))
}

-- sp-doc-specification
id-spq-ets-docspec OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4)
    etsi(0) cades(19122) id-spq (2) 1 }

SPDocSpecification ::= CHOICE {
    oid OBJECT IDENTIFIER,
    uri IA5String
}

```

```

-- signature-policy-store attribute (clause 5.2.10)
id-aa-ets-sigPolicyStore OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4)
    etsi(0) cadres(19122) attributes(1) 3 }

SignaturePolicyStore ::= SEQUENCE {
    spDocSpec      SPDocSpecification ,
    spDocument     SignaturePolicyDocument
}

SignaturePolicyDocument ::= CHOICE {
    sigPolicyEncoded  OCTET STRING,
    sigPolicyLocalURI IA5String
}

-- signature-timestamp attribute (clause 5.3)
id-aa-signatureTimeStampToken OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 14 }

SignatureTimeStampToken ::= TimeStampToken

-- Archive validation data
-- =====

-- ats-hash-index-v3 attribute (clause 5.5.2)
id-aa-ATSHashIndex-v3 OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4)
    etsi(0) cadres(19122) attributes(1) 5 }

ATSHashIndexV3 ::= SEQUENCE {
    hashIndAlgorithm      AlgorithmIdentifier,
    certificatesHashIndex SEQUENCE OF OCTET STRING,
    crlsHashIndex         SEQUENCE OF OCTET STRING,
    unsignedAttrValuesHashIndex SEQUENCE OF OCTET STRING
}

-- archive-time-stamp-v3 attribute (clause 5.5.3)
id-aa-ets-archiveTimeStampV3 OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4)
    etsi(0) electronic-signature-standard(1733) attributes(2) 4 }

ArchiveTimeStampToken ::= TimeStampToken

-- Additional attributes for validation data
-- =====

-- complete-certificate-references attribute (clause A.1.1.1)
id-aa-ets-certificateRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 21 }

CompleteCertificateRefs ::= SEQUENCE OF OtherCertID

OtherCertID ::= SEQUENCE {
    otherCertHash OtherHash,
    issuerSerial  IssuerSerial OPTIONAL
}

OtherHash ::= CHOICE {
    sha1Hash OtherHashValue, -- This contains a SHA-1 hash
    otherHash OtherHashAlgAndValue
}

-- certificate-values attribute (clause A.1.1.2)
id-aa-ets-certValues OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 23 }

CertificateValues ::= SEQUENCE OF Certificate

-- complete-revocation-references attribute (clause A.1.2.1)

```

```

id-aa-ets-revocationRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 22 }

CompleteRevocationRefs ::= SEQUENCE OF CrlOcspRef

CrlOcspRef ::= SEQUENCE {
  crlids [0] CRLListID OPTIONAL,
  ocspids [1] OcspListID OPTIONAL,
  otherRev [2] OtherRevRefs OPTIONAL
}
CRLListID ::= SEQUENCE {
  crls SEQUENCE OF CrlValidatedID
}

CrlValidatedID ::= SEQUENCE {
  crlHash OtherHash,
  crlIdentifier CrlIdentifier OPTIONAL
}

CrlIdentifier ::= SEQUENCE {
  crlissuer Name,
  crlIssuedTime UTCTime,
  crlNumber INTEGER OPTIONAL
}

OcspListID ::= SEQUENCE {
  ocspResponses SEQUENCE OF OcspResponsesID
}

OcspResponsesID ::= SEQUENCE {
  ocspIdentifier OcspIdentifier,
  ocspRefHash OtherHash OPTIONAL
}

OcspIdentifier ::= SEQUENCE {
  ocspResponderID ResponderID, -- As in OCSF response data
  producedAt GeneralizedTime -- As in OCSF response data
}

OtherRevRefs ::= SEQUENCE {
  otherRevRefType OTHER-REVOCATION-REF.&id,
  otherRevRefs SEQUENCE OF OTHER-REVOCATION-REF.&Type
}

OTHER-REVOCATION-REF ::= CLASS {
  &Type,
  &id OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
  WITH SYNTAX &Type ID &id }

-- certificate-revocation-values attribute (clause A.1.2.2)

id-aa-ets-revocationValues OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 24 }

RevocationValues ::= SEQUENCE {
  crlVals [0] SEQUENCE OF CertificateList OPTIONAL,
  ocspVals [1] SEQUENCE OF BasicOCSPResponse OPTIONAL,
  otherRevVals [2] OtherRevVals OPTIONAL
}

OtherRevVals ::= SEQUENCE {
  otherRevValType OTHER-REVOCATION-VAL.&id,
  otherRevVals SEQUENCE OF OTHER-REVOCATION-REF.&Type
}

OTHER-REVOCATION-VAL ::= CLASS {
  &Type,
  &id OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX {
  WITH SYNTAX &Type ID &id }

-- attribute-certificate-references attribute (clause A.1.3)

id-aa-ets-attrCertificateRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 44 }

```

```

AttributeCertificateRefs ::= SEQUENCE OF OtherCertID

-- attribute-revocation-references attribute (clause A.1.4)

id-aa-ets-attrRevocationRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 45 }

AttributeRevocationRefs ::= SEQUENCE OF CrlOcspRef

-- time-stamped-certs-crls-references attribute (clause A.1.5.1)

id-aa-ets-certCRLTimestamp OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 26 }

TimestampedCertsCRLs ::= TimeStampToken

-- CADES-C-timestamp attribute (clause A.1.5.2)

id-aa-ets-escTimeStamp OBJECT IDENTIFIER ::= { iso(1) member-body(2)
  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 25 }

ESCTimeStampToken ::= TimeStampToken

END

```

The following module was added in v1.2.1.

```

ETSI-CADES-19122v121 { itu-t(0) identified-organization(4) etsi(0) cades(19122)
  id-mod(0) cades-19122v121(2)}

DEFINITIONS EXPLICIT TAGS ::=
BEGIN
EXPORTS All;

IMPORTS

-- Imports from Cryptographic Message Syntax (CMS) Algorithm Identifier Protection Attribute:
-- IETF RFC 6211
  id-aa-CMSAlgorithmProtection, CMSAlgorithmProtection
  FROM CMSAlgorithmProtectionAttribute
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) modules(0)
    id-mod-cms-algorithmProtect(52) }

;

-- Object Identifier arc signed assertions within the signer-attribute-v2
id-etsi-cades-spq OBJECT IDENTIFIER ::=
  { itu-t(0) identified-organization(4) etsi(0) cades(19122) signed-assertions(3) }

-- signed-SAML-assertion
-- =====
id-ets-signedSAML OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4)
  etsi(0) cades(19122) signed-assertions (3) 0 }

SignedSAMLAssertion ::= OCTET STRING

END

```

---

## Annex E (informative): Example Structured Contents and MIME

### E.1 Use of MIME to Encode Data

#### E.1.1 MIME Structure

The signed content may be structured using MIME (Multipurpose Internet Mail Extensions - IETF RFC 2045 [2]). Whilst the MIME structure was initially developed for Internet email, it has a number of features that make it useful to provide a common structure for encoding a range of electronic documents and other multi-media data (e.g. photographs, video). These features include:

- providing a means of signalling the type of "object" being carried (e.g. text, image, ZIP file, application data);
- providing a means of associating a file name with an object;
- associating several independent objects (e.g. a document and image) to form a multi-part object;
- handling data encoded in text or binary and, if necessary, re-encoding the binary as text.

When encoding a single object, MIME consists of:

- header information; followed by
- encoded content.

This structure can be extended to support multi-part content.

#### E.1.2 Header Information

A MIME header includes:

- MIME Version information:  
e.g.: `MIME-Version: 1.0`
- Content type information, which includes information describing the content sufficient for it to be presented to a user or application process, as required. This includes information on the "media type" (e.g. text, image, audio) or whether the data is for passing to a particular type of application. In the case of text, the content type includes information on the character set used.  
e.g. `Content-Type: text/plain; charset="us-ascii"`
- Content-encoding information, which defines how the content is encoded (see below about encoding supported by MIME).
- Other information about the content, such as a description or an associated file name.

An example MIME header for text object is:

```
Mime-Version: 1.0
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: quoted-printable
```

An example MIME header for a binary file containing a PDF document is:

```
Content-Type: application/pdf
Content-Transfer-Encoding: base64
Content-Description: JCFV201.pdf
Content-Disposition: filename="JCFV201.pdf"
```

## E.1.3 Content Encoding

MIME supports a range of mechanisms for encoding both text and binary data.

Text data can be carried transparently as lines of text data encoded in 7- or 8-bit ASCII characters. MIME also includes a "quoted-printable" encoding that converts characters other than the basic ASCII into an ASCII sequence.

Binary can either be carried:

- transparently as 8-bit octets; or
- converted to a basic set of characters using a system called Base64.

NOTE: As there are some mail relays that can only handle 7-bit ASCII, Base64 encoding is usually used on the Internet.

## E.1.4 Multi-Part Content

Several objects (e.g. text and a file attachment) can be associated together using a special "multi-part" content type. This is indicated by the content type "multipart" with an indication of the string to be used indicating a separation between each part.

In addition to a header for the overall multipart content, each part includes its own header information indicating the inner content type and encoding.

An example of a multipart content is:

```
Mime-Version: 1.0
Content-Type: multipart/mixed; boundary="-----_NextPart_000_01BC4599.98004A80"
Content-Transfer-Encoding: 7bit

-----_NextPart_000_01BC4599.98004A80
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: 7bit

Per your request, I've attached our proposal for the Java Card Version
2.0 API and the Java Card FAQ.

-----_NextPart_000_01BC4599.98004A80
Content-Type: application/pdf; name="JCFV201.pdf"
Content-Transfer-Encoding: base64
Content-Description: JCFV201.pdf
Content-Disposition: attachment; filename="JCFV201.pdf"

OM8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAAAAAAACAAAAAgAAAAAAAAAA
EAAAtAAAAEAAAD+///AAAAAMAAAAGAAAAA////////////////////
AANhAAQAYg==

-----_NextPart_000_01BC4599.98004A80--
```

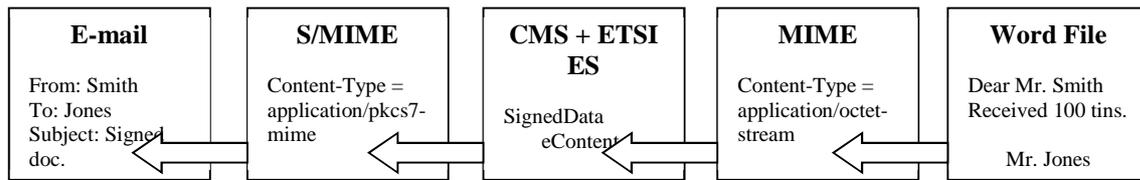
Multipart content can be nested. So a set of associated objects (e.g. HTML text and images) can be handled as a single attachment to another object (e.g. text).

The Content-Type from each part of the MIME message indicates the type of content.

## E.2 S/MIME

### E.2.1 Using S/MIME

The specific use of MIME to carry CMS (extended as defined in the present document) secured data is called S/MIME (see IETF RFC 3851 [i.14]).



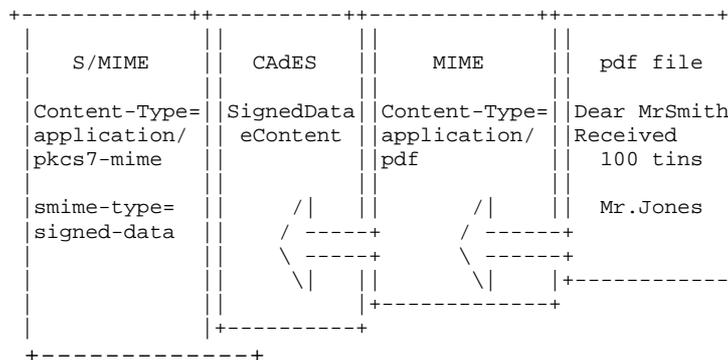
**Figure E.1: Illustration of relation of using S/MIME**

S/MIME carries digital signatures as either:

- an "application/pkcs7-mime" object with the CMS carried as binary attachment (PKCS7 is the name of the early version of CMS), see clause E.2.2; or
- a "multipart/signed" object with the signed data and the signature encoded as separate MIME objects.

## E.2.2 Using application/pkcs7-mime

The data to be signed can be included in the SignedData within CADES, which itself can be included in a single S/MIME object. See IETF RFC 3851 [i.14], clause 3.4.2 and figure E.2.



**Figure E.2: Signing Using application/pkcs7-mime**

This approach is similar to handling signed data as any other binary file attachment.

An example of signed data encoded using this approach is:

```
Content-Type: application/pkcs7-mime; smime-type=signed-data;
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
```

```
567GhIGfHfYT6ghyHhHUujpFyF4f8HHGTrfvhJhjH776tbB9HG4VQbnj7
77n8HHGT9HG4VQpFyF467GhIGfHfYT6rfvbnj756tbBghyHhHUujhJhjH
HUujhJh4VQpFyF467GhIGfHfYGT6rfvbnjT6jH7756tbB9H7n8HHGghyHh
6YT64V0GhIGfHfQbnj75
```

## E.2.3 Using multipart/signed and application/pkcs7-signature

The signed data is not included in the SignedData, and the CMS structure only includes the signature. See IETF RFC 3851 [i.14], clause 3.4.3 and figure E.3.

CMS also supports an alternative structure where the signature and data being protected are separate MIME objects carried within a single message. In this case, the data to be signed is not included in the SignedData, and the CMS structure only includes the signature. See IETF RFC 3851 [i.14], clause 3.4.3 and figure E.3 hereafter. In this case a multipart/signed message is used, where the signature is included as application/pkcs7-signature.

An example of signed data encoded using this approach is:

```
Content-Type: multipart/signed;
```

```

protocol="application/pkcs7-signature";
micalg=shal; boundary=boundary42

--boundary42
Content-Type: text/plain

This is a clear-signed message.

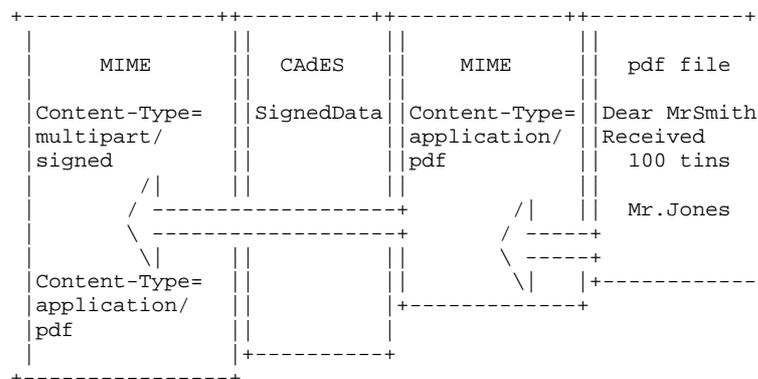
--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

ghyHhHUujhJh77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jh77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJh776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpFyF4
7GhIGfHfYT64VQbnj756

--boundary42--

```

With this second approach, the signed data passes through the CMS process and is carried as part of a multiple-parts signed MIME structure, as illustrated in figure E.3. The CMS structure just holds the digital signature.



**Figure E.3: Signing Using application/pkcs7-signature**

This second approach (multipart/signed) has the advantage that the signed data can be decoded by any MIME-compatible system even if it does not recognize CMS-encoded digital signatures.

## E.3 Use of MIME in the signature

CADES allows two ways to include the MIME type of the data to be signed, either in the `contentDescription` element of the `content-hints` attribute or in the `mime-type` attribute. The included MIME type allows to give information on how the driving application should decode or display the signed data. Thus these attributes allow to give the application useful information. In addition, including the MIME type into the signature can also help to prevent attacks based on the fact that a binary data file might change its meaning/use depending on the application used to process the data.

Which information of the MIME header is included as MIME type into the signature depends on the needs of the driving application. Two examples follow.

- 1) For most applications, it will be sufficient to know the application corresponding to signed data, thus they will put only the application or only the Content-Type as MIME-type into the signature, for example:
  - `application/pdf`; or
  - `text/plain; charset="us-ascii"`.
- 2) In the case that the driving application is interested in all the details of the MIME header, it can put the whole header as MIME-type into the signature, like for example:

```

Content-Type: application/pdf
Content-Transfer-Encoding: base64

```

Content-Description: JCFV201.pdf  
Content-Disposition: filename="JCFV201.pdf"

## Annex F (informative): Change History

Date	Version	Information about changes
October 2019	1.1.2	Implementation of change request ESI(19)68_052r3: Specification of CAdES zero policy hash
October 2020	1.1.3	Implementation of change request ESI(20)071_021r1: clarification on computation of unsignedAttrValuesHashIndex field of ats-hash-index-v3 attribute
November 2020	1.1.4	Implementation of change request ESI(20)071_021r2: clarification on computation of unsignedAttrValuesHashIndex field of ats-hash-index-v3 attribute as accepted during ESI#71
May 2021	1.1.5	Implementation of <ul style="list-style-type: none"> <li>• CR#1 order of hash index: ESI(21)072014r1</li> <li>• CR#2 inclusion of mime type: ESI(21)072015r1</li> <li>• CR#3 clarification on usage of countersignatures: ESI(21)072016r1</li> <li>• CR#4 OID for SAMLv2 to be included in signer-attributes-v2: ESI(21)072017r2</li> <li>• CR#5 include signed attribute to protect algorithm used: ESI(21)072018r2</li> <li>• CR#6 Reference to specific version of legacy documents: ESI(21)072051</li> <li>• CR#7 remove informative ASN.1 annex: ESI(21)072060</li> <li>• CR#8 less restrictions on CMS Version: ESI(21)073017</li> </ul>
July 2021	1.1.6	Adding of the change history

## History

<b>Document history</b>		
V1.0.1	July 2015	Publication as ETSI TS 119 122-1 (Withdrawn)
V1.1.1	April 2016	Publication
V1.1.5	July 2021	EN Approval Procedure AP 20211017: 2021-07-19 to 2021-10-18
V1.2.1	October 2021	Publication