Draft ETSI TS 119 182-1 V0.0.2 (2020-01)

Electronic Signatures and Infrastructures (ESI);

JAdES digital signatures built on JSON Web Signatures;

Part 1: Building blocks and JAdES baseline signatures;

Author: Juan Carlos Cruellas (cruellas@ac.upc.edu)

Note: this is a major revision to the earlier version taking an approach defining requirements bottom up starting from JWS RFC 7515

Disclosure and distribution of this document is allowed only to ETSI ESI TC members and OBE members.

<

**TECHNICAL SPECIFICATION**

Reference

<Workitem>

Keywords

<keywords>

***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 - NAF 742 C

Association à but non lucratif enregistrée à la

Sous-préfecture de Grasse (06) N° 7803/88

***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 only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

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/CommiteeSupportStaff.aspx>

***Copyright Notification***

Reproduction is only permitted for the purpose of standardization work undertaken within ETSI.

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

© ETSI yyyy.

All rights reserved.

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

Contents

[Intellectual Property Rights 5](#_Toc30419350)

[Foreword 5](#_Toc30419351)

[Modal verbs terminology 5](#_Toc30419352)

[1 Introduction 5](#_Toc30419353)

[1 Scope 6](#_Toc30419354)

[2 References 7](#_Toc30419355)

[2.1 Normative references 7](#_Toc30419356)

[2.2 Informative references 8](#_Toc30419357)

[3 Definitions, abbreviations and terminology 8](#_Toc30419358)

[3.1 Definitions 8](#_Toc30419359)

[3.2 Abbreviations 9](#_Toc30419360)

[3.3 Terminology 9](#_Toc30419361)

[4 General Requirements 10](#_Toc30419362)

[5 Header parameters semantics and syntax 11](#_Toc30419363)

[5.1 Use of header parameters defined in RFC 7515 11](#_Toc30419364)

[5.1.1 Introduction 11](#_Toc30419365)

[5.1.2 The alg (X.509 URL) header parameter 11](#_Toc30419366)

[5.1.3 The cty (content type) header parameter 11](#_Toc30419367)

[5.1.4 The kid (key identifier) header parameter 11](#_Toc30419368)

[5.1.5 The x5u (X.509 URL) header parameter 12](#_Toc30419369)

[5.1.6 The x5t (X.509 Certificate SHA-1 Thumbprint) header parameter 12](#_Toc30419370)

[5.1.7 The x5t#S256 (X.509 Certificate SHA-256 Thumbprint) header parameter 12](#_Toc30419371)

[5.1.8 The x5c (X.509 Certificate Chain) header parameter 12](#_Toc30419372)

[5.1.9 The crit (critical) header parameter 13](#_Toc30419373)

[5.2 New signed header parameters 13](#_Toc30419374)

[5.2.1 The sigT (claimed signing time) header parameter 13](#_Toc30419375)

[5.2.2 The x5t#o (X509 certificate digest) header parameter 13](#_Toc30419376)

[5.2.3 The srCm (signer commitment) header parameter 14](#_Toc30419377)

[5.2.4 The sigPl (signature production place) header parameter 15](#_Toc30419378)

[5.2.5 The srAts (signer attributes) header parameter 15](#_Toc30419379)

[5.2.6 The adoTst (signed data time-stamp) header parameter 17](#_Toc30419380)

[5.2.7 The sigPId (signature policy identifier) header parameter 17](#_Toc30419381)

[5.2.7.1 Semantics and syntax 17](#_Toc30419382)

[5.2.7.2 Signature policy qualifiers 18](#_Toc30419383)

[5.2.8 The sigD header parameter 19](#_Toc30419384)

[5.2.8.1 Semantics and Syntax 19](#_Toc30419385)

[5.2.8.2 Mechanism HttpHeaders 20](#_Toc30419386)

[5.2.8.3 Mechanism ObjectIdByURI 21](#_Toc30419387)

[5.2.8.4 Mechanism ObjectIdByURIHash 21](#_Toc30419388)

[5.3 New unsigned header parameter 21](#_Toc30419389)

[5.3.1 The etsiU header parameter 21](#_Toc30419390)

[5.3.2 The cSig (counter signature) JSON object 23](#_Toc30419391)

[5.3.3 The sigPSt JSON object 23](#_Toc30419392)

[5.3.4 The sigTst JSON object 24](#_Toc30419393)

[5.3.5 JSON objects for validation data values 25](#_Toc30419394)

[5.3.5.1 The xVals JSON array 25](#_Toc30419395)

[5.3.5.2 The rVals JSON object 25](#_Toc30419396)

[5.3.5.3 The axVals JSON array 27](#_Toc30419397)

[5.3.5.4 The arVals JSON object 27](#_Toc30419398)

[5.3.6 JSON values for long term availability and integrity of validation material 27](#_Toc30419399)

[5.3.6.1 The tstVd JSON object 27](#_Toc30419400)

[5.3.6.2 The arcTst JSON object 28](#_Toc30419401)

[5.3.6.3 Computation of message-imprint 29](#_Toc30419402)

[5.3.6.3.1 Time-stamping the time-stamp token of the last arcTst 29](#_Toc30419403)

[5.3.6.3.2 Time-stamping all the contents of the JAdES signature 29](#_Toc30419404)

[1.1.1.1.1.1 Base64url incorporation 29](#_Toc30419405)

[1.1.1.1.1.2 Clear JSON incorporation 30](#_Toc30419406)

[5.3.7 The sigPSt JSON object 31](#_Toc30419407)

[5.4 Generally useful syntax 31](#_Toc30419408)

[5.4.1 The oId data type 31](#_Toc30419409)

[5.4.2 The pkiOb data type 32](#_Toc30419410)

[5.4.3 Container for electronic time‑stamps 33](#_Toc30419411)

[5.4.3.1 Introduction 33](#_Toc30419412)

[5.4.3.2 Containers for electronic time-stamps 33](#_Toc30419413)

[5.4.3.3 The tstContainer type 33](#_Toc30419414)

[6 JAdES baseline signatures 34](#_Toc30419415)

[6.1 Signature levels 34](#_Toc30419416)

[6.2 General requirements 35](#_Toc30419417)

[6.2.1 Algorithm requirements 35](#_Toc30419418)

[6.2.2 Notation for requirements 35](#_Toc30419419)

[6.3 Requirements on JAdES components and services 38](#_Toc30419420)

[Annex A (normative): Additional header parameters Specification 43](#_Toc30419421)

[A.1 Header parameters for validation data 43](#_Toc30419422)

[A.1.1 The xRefs header parameter 43](#_Toc30419423)

[A.1.2 The rRefs header parameter 44](#_Toc30419424)

[A.1.3 The axRefs header parameter 46](#_Toc30419425)

[A.1.4 The arRefs header parameter 47](#_Toc30419426)

[A.1.5 Time‑stamps on references to validation data 48](#_Toc30419427)

[A.1.5.1 The sigRTst header parameter 48](#_Toc30419428)

[A.1.5.1.2 Computation of the message imprint with Base64url incorporation 48](#_Toc30419429)

[A.1.5.1.3 Computation of the message imprint with JSON clear incorporation 48](#_Toc30419430)

[A.1.5.2 The rfsTst header parameter 49](#_Toc30419431)

[A.1.5.2.1 Semantics and syntax 49](#_Toc30419432)

[A.1.5.2.2 Computation of the message imprint with Base64url incorporation 49](#_Toc30419433)

[A.1.5.2.3 Computation of the message imprint with clear JSON incorporation 49](#_Toc30419434)

[Annex B (normative): JSON Schema file 50](#_Toc30419435)

[B.1 JSON Schema file location for JAdES header parameters 50](#_Toc30419436)

[Annex C (informative): Correspondence between XAdES tags and JAdES tags 50](#_Toc30419437)

[C.1 Correspondence between XAdES qualifying properties tags and JAdES header parameter tags 50](#_Toc30419438)

[Annex D (normative): Alternative mechanisms for long term availability and integrity of validation data 52](#_Toc30419439)

[Annex E (informative): Bibliography 53](#_Toc30419440)

[Annex F (informative): Change History 54](#_Toc30419441)

[History 55](#_Toc30419442)

# ­Intellectual Property Rights

# Foreword

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

**Part 1: "JAdES signatures built on JWS: JAdES"**

**Subpart 1: "Building blocks and JAdES baseline signatures";**

Subpart 2: "Extended JAdES signatures".

One JSON schema file, whose location is detailed in clause B.1 and which contain JSON Schema definitions complements the present document.

# 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](https://portal.etsi.org/Services/editHelp%21/Howtostart/ETSIDraftingRules.aspx) (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.1].

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

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

# Scope

The present document:

1. Specifies a JSON [1] format for AdES signatures (JAdES signatures hereinafter) built on JSON Web Signatures (JWS hereinafter) as specified in IETF RFC 7515: "JSON Web Signature (JWS)" [2]. For this, the present specification:
* Extends the JSON Web Signatures (JWS hereinafter) specified in IETF RFC 7515: "JSON Web Signature (JWS)" [2] by defining an additional set of JSON header parameters that can be incorporated in the JOSE Header (either in its JWS Protected Header or its JWS Unprotected Header parts). Many of these new header parameters have the same semantics as the attributes/properties defined in CAdES [i.2] and XAdES [5] digital signatures. Other header parameters are defined to meet specific requirements that current JSON Web Signatures cannot meet (e.g. for identifying data objects to be included within the signature). These new header parameters and their corresponding types are defined in a JSON schema.
* Specifies the mechanisms for incorporating the aforementioned JSON components in JSON Web Signatures [2] to build JAdES signatures, offering the same features as CAdES and XAdES in JSON syntax, and therefore fulfilling the same requirements (such as the long-term validity of digital signatures).
1. Defines four levels of JAdES baseline signatures addressing incremental requirements to maintain the validity of the signatures over the long term. Each level requires the presence of certain JAdES header parameters, suitably profiled for reducing the optionality as much as possible. The aforementioned levels provide the basic features necessary for a wide range of business and governmental use cases for electronic procedures and communications to be applicable to a wide range of communities when there is a clear need for interoperability of digital signatures used in electronic documents

EXAMPLE: An example of requirements raised in specific domains is signing HTTP messages exchanged by parties in certain environments, which require signing both the HTTP payload and some specific http headers. The format specified in IETF RFC 7515 [2] does not provide any native mechanism for individually identifying signed data objects, as the JWS Payload is a unique sequence of octets. Clause 5.2.8 of the present document defines sigD, a new JSON header parameter that allows to identify one or more individual data objects that contribute to the computation of the JWS Signature Value and consequently, are signed.

NOTE 1: IETF RFC 7515: "JSON Web Signature (JWS)" [2] defines a structure (the JSON object signatures) that allows parallel signatures on a single payload. The JAdES signatures specified in the present document can use this structure for signing multiple data objects with a single signature.

Procedures for creation, augmentation, and validation of JAdES digital signatures are out of scope.

NOTE 2: ETSI EN 319 102‑1 [i.3] specifies procedures for creation, augmentation and validation of other types of AdES digital signatures.

The present multi-part Technical Specification aims at supporting electronic signatures in different regulatory frameworks.

NOTE 3: Specifically, but not exclusively, JAdES digital signatures specified in the present multi-part Technical Specification 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.1].

# References

## 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 <http://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] IETF RFC 8259: "The JavaScript Object Notation (JSON) Data Interchange Format" (December 2017).

[2] IETF RFC 7515: "JSON Web Signature (JWS)" (May 2015).

[3] IETF RFC 6901: " The JavaScript Object Notation (JSON) Pointer" (April 2013).

[4] IETF RFC 3061: "A URN Namespace of Object Identifiers".

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

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

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

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

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

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

[11] IETF RFC 5816: "ESSCertIDv2 Update for RFC 3161".

[12] IETF RFC 1779: "A String Representation of Distinguished Names".

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

[14] IETF RFC 3230: "Instance Digests in HTTP". January 2002.

[15] IETF RFC 7797: "JSON Web Signature (JWS) Unencoded Payload Option". February 2016.

[16] IETF RFC 3339: "Date and Time on the Internet: Timestamps". July 2002.

[17] draft-cavage-http-signatures-10: "Signing HTTP Messages". May 2018.

EDITOR’S REQUEST FOR ADVICING: This document is a draft, and usually a TS can not have a draft as a normative reference; advice is requested as to how to manage this situation.

[18] IETF RFC 3230: "Instance Digests in HTTP". January 2002.

## 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] Regulation (EU) No 910/2014 of the European Parliament and of the Council on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC. OJ L 257, 28.08.2014, p. 73-114.

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

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

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

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

[i.6] ETSI TR 119 100: "Electronic Signatures and Infrastructures (ESI); Business Driven Guidance for Signature Creation and Validation".

[i.7] W3C Recommendation: "XML Signature Syntax and Processing. Version 1.1" (11 April 2013).

[i.8] 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.9] OASIS Standard: "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0".

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

[i.12] ETSI TS 119 312: "Electronic Signatures and Infrastructures (ESI); Cryptographic Suites".

[i.13] IETF RFC 6931: "Additional XML Security Uniform Resource Identifiers (URIs)".

[i.14] ETSI TS 119 182-2: " Electronic Signatures and Infrastructures (ESI); JAdES digital signatures; JAdES digital Signatures built on JSON Web Signatures: JAdES; Part 2: Extended JAdES signatures". It can be found at: <https://tools.ietf.org/html/draft-cavage-http-signatures-10>.

[i.15] ETSI TS 119 312: "Electronic Signatures and Infrastructures (ESI); Cryptographic Suites" v1.3.1

# Definitions, abbreviations and terminology

## Definitions

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

**Base64url:** base64 encoding that uses "the URL- and filename-safe character set defined in Section 5 of IETF RFC 4648: "The Base16, Base32, and Base64 Data Encodings" [13], with all trailing '=' characters omitted (as permitted by Section 3.2 of IETF RFC 4648 [13]) and without the inclusion of any line breaks, whitespace, or other additional characters.

NOTE: Definition taken from IETF RFC 7515 [2]. Appendix C of this document provides notes on implementing base64url without padding.

**Data to be signed:** the data used as input to the creation of the JAdES signature.

**JAdES signature:** a JSON Web Signature as specified in IETF RFC 7515 [2], which meets the requirements specified within the present document or in ETSI TS 119 182-2 [i.14].

**JSON Web Signature**: A data structure representing a digitally signed message as defined in RFC 7515

**JWS Signature Value:** the digital signature cryptographic value calculated over a sequence of octets derived from the JWS Protected Header and data to be signed.

## Abbreviations

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

CA Certification Authority

CD European Commission Decision

CRL Certificate Revocation List

JSON Javascript Object Notation

JWS JSON Web Signature

HTTP Hyper Text Transfer Protocol

OCSP Online Certificate Status Protocol

OID Object IDentifier

PKI Public Key Infrastructure

SAML Security Assertion Markup Language

SPO Service Provision Option

TSA Time‑Stamping Authorities

TSL Trust-service Status List

TSP Trusted Service Providers

TSU Time-Stamping Unit

URI Uniform Resource Identifier

URL Uniform Resource Locator

URN Uniform Resource Name

UTC Coordinated Universal Time

## Terminology

The present document adopts the same terminology for naming the different components of JSON Web Signatures. Of special relevance is to notice that the term "JSON Web Signature" shall denote the JSON structure specified in IETF RFC 7515 [2].

The present document uses the term "JSON value" for denoting JSON objects, or JSON arrays, or JSON numbers, or JSON strings, i.e. a subset of the potential meanings of "JSON value" listed in clause 3 of IETF RFC 8259: "The JavaScript Object Notation (JSON) Data Interchange Format" [1].

The present document uses the term "header parameter" for denoting a JSON object, JSON array, JSON number, or JSON string, which is member either of the JWS Protected Header or the JWS Unprotected Header specified in IETF RFC 8259 [1].

The present document uses the term "member" for denoting a JSON object's member, as specified in clause 4 of IETF RFC 8259: "The JavaScript Object Notation (JSON) Data Interchange Format" [1].

The present document uses the term "element" or "element of the array" for denoting the contents of a position within a JSON array (specified in clause 5 of IETF RFC 8259 [1]).

NOTE: These last terms will be used for denoting each of the JSON values that shall be added to the etsiU JSON array, which will be incorporated in the JWS Unprotected header as a header parameter, so that the aforementioned JSON values will play within the present document an equivalent role to the role played by the unsigned attributes in CAdES and the unsigned qualifying properties in XAdES.

The present document uses the term "JAdES component" for denoting any JAdES signature constituent, namely a header parameter, the JWS Payload, or the JWS Signature.

As for the names of the header parameters, the following criteria and conventions have been used:

1. The names have been selected to have a maxim length of 8 characters; most of the names are shorter.
2. The names of header parameters qualifying the signature itself use to start with "sig".
3. The names of header parameters qualifying the signer use to start with "sr".
4. The names of header parameters qualifying the data to be signed use to start with "sd".
5. The names of header parameters dealing with time-stamp tokens use to finalize with "tst".
6. The names of header parameters dealing with certificates use to start or contain "x" (following the convention of IETF RFC 7515 [2], which defines the header parameters x5u, x5c, x5t, and x5t#S256).
7. The names of header parameters dealing with revocation values (CRLs or OCSP responses) use to start or contain "r".
8. The names of header parameters dealing with attribute certificates or the corresponding revocation values use to start "a".
9. The names of header parameters dealing with values (of certificates or revocation values) use to contain "Vals".
10. The names of header parameters dealing with references (to certificates or revocation values) use to contain "Refs" (except x5t, and x5t#S256, which have been defined in IETF RFC 7515 [2], contain references to certificates, and do not include it).

# General Requirements

The JAdES components defined in the present document shall be carried within the JOSE header as specified in IETF RFC 7515 [2].

All the JAdES signed header parameters specified in clause 5.2 of the present document, as well as: cty, kid, and x5u header parameters specified in IETF RFC 7515 [2] and further profiled in clause 5.1 of the present document, shall be incorporated as header parameters of the JWS Protected Header of the JSON Web Signature, specified in IETF RFC 7515 [2].

The JWS Protected header may be encoded using either JWS Compact Serialization or JWS JSON Serialization as specified in section 3 of IETF RFC 7515 [2]

EDITOR’S NOTE: THE REFERENCE TO THE DOCUMENTS SPECIFYING THE JSON SCHEMA SHALL BE ADDED AFTER SOLVING THE ISSUE OF NORMATIVELY REFERENCING IETF DRAFTS, AS THEY ARE DRAFTS, OR WHEN THEY ADVANCE IN THE STANDARDIZATION PROCESS.

One JSON array, namely the etsiU header parameter shall be incorporated within the JWS Unprotected Header of the JSON Web Signature.

The elements of this JSON array shall contain JSON values that play for JAdES signatures the same role as the role played by the unsigned attributes for CAdES signatures, and the role played by the unsigned qualifying properties for XAdES signatures. The etsiU header parameter and its elements are specified in clause 5.3.

All the header parameters

NOTE 1: At the moment of producing the present document, JSON Schema was under development. The working draft was 07. The corresponding documents can be found at: <https://json-schema.org/specification.html>.

NOTE 2: Although at the moment of producing the present document there exist several proposals for JSON canonicalization algorithms, none have been formally adopted by any standardisation organisation. Nevertheless, the present document uses placeholders for identifiers of canonicalization algorithms in a number of components that could use them if such algorithms are standardized in the future.

# Header parameters semantics and syntax

## Use of header parameters defined in RFC 7515

### Introduction

This clause defines additional requirements for some of header parameters specified in IETF RFC 7515 [2].

JAdES signatures may incorporate any of the header parameters specified in IETF RFC 7515 [2] and IETF RFC 7797 [15].

NOTE Clause 6.3 specifies requirements for some of the header parameters specified in IETF RFC 7515 [2] for JAdES baseline signatures.

### The alg (X.509 URL) header parameter

**Semantics**

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

The alg header parameter shall have the semantics specified in IETF RFC 7515 [2], clause 4.1.1.

**Syntax**

The alg header parameter shall have the syntax specified in IETF RFC 7515 [2], clause 4.1.1.

Its value should be one of the algorithms for digital signatures recommended by in ETSI TS 119 312: "Electronic Signatures and Infrastructures (ESI); Cryptographic Suites" v1.3.1[i.15].

### The cty (content type) header parameter

**Semantics**

The cty header parameter shall be a signed header parameter that qualifies the signed data object.

The cty header parameter shall have the semantics specified in IETF RFC 7515 [2], clause 4.1.10.

The cty header parameter shall not be present if the sigD header parameter, specified in clause 5.2.8 of the present document, is present within the JAdES signature.

The cty header parameter shall not be present if the content type is implied by the signed data object.

The cty header parameter shall not be present if the signed data object is a (counter-signed) signature.

NOTE: The sigD header parameter has one member that contains information of the format and type of the signed data objects.

**Syntax**

The cty header parameter shall have the syntax specified in IETF RFC 7515 [2], clause 4.1.10.

### The kid (key identifier) header parameter

**Semantics**

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

The kid header parameter shall have the semantics specified in IETF RFC 7515 [2], clause 4.1.4.

The content of kid header parameter shall be the base64 encoding of one DER-encoded instance of type IssuerSerial type defined in IETF RFC 5035 [6].

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

**Syntax**

The kid header parameter shall have the syntax specified in IETF RFC 7515 [2], clause 4.1.4.

### The x5u (X.509 URL) header parameter

**Semantics**

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

The x5u header parameter shall have the semantics specified in IETF RFC 7515 [2], clause 4.1.5.

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

**Syntax**

The x5u header parameter shall have the syntax specified in IETF RFC 7515 [2], clause 4.1.5.

### The x5t (X.509 Certificate SHA-1 Thumbprint) header parameter

The x5t component, specified in clause 4.1.7 of IETF RFC 7515 [2] shall not be present within a JAdES signature neither as a header parameter within the JWS Protected Header, nor as an element of the the etsiU unsigned header parameter (specified in clause 5.3.1 of the present document).

### The x5t#S256 (X.509 Certificate SHA-256 Thumbprint) header parameter

**Semantics**

The x5t#S256 shall be a signed header parameter that qualifies the signature.

The x5t#S256 header parameter shall have the semantics specified in IETF RFC 7515 [2], clause 4.1.8.

**Syntax**

The x5t#S256 header parameter shall have the syntax specified in IETF RFC 7515 [2], clause 4.1.8.

If header parameters x5c (specified in clause 4.1.6 of IETF RFC 7515 [2]) and x5t#o (specified in clause 5.2.2 of the present document) are absent, then the x5t#S256 header parameter shall be incorporated in the JWS Protected Header.

If the x5t#S256 header parameter is incorporated in the JWS Protected Header, the x5t#o header parameter (specified in clause 5.2.2 of the present document) shall be absent.

If the x5t#S256 header parameter is incorporated in the JWS Protected Header, then the x5c may be absent.

### The x5c (X.509 Certificate Chain) header parameter

**Semantics**

The x5c header parameter maybe either a signed header parameter or an element of the etsiU unsigned header parameter that qualifies the signature.

The x5c header parameter shall have the semantics specified in IETF RFC 7515 [2], clause 4.1.6.

**Syntax**

The x5c header parameter shall have the syntax specified in IETF RFC 7515 [2], clause 4.1.6.

If header parameters x5t#S256 (specified in clause 4.1.8 of IETF RFC 7515 [2]) and x5t#o (specified in clause 5.2.2 of the present document) are absent, then the x5c header parameter shall be incorporated in the JWS Protected Header.

If the x5c header parameter is incorporated in the JWS Protected Header, then the header parameters that contain the digest value of the signing certificate, namely the x5t#S256 (specified in clause 4.1.8 of IETF RFC 7515 [2]) and the x5t#o (specified in clause 5.2.2 of the present document) may be absent.

### The crit (critical) header parameter

**Semantics**

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

The crit header parameter shall have the semantics specified in IETF RFC 7515 [2], clause 4.1.11.

The JAdES signatures incorporating any signed header parameter specified in clauses New signed header parameters5.2 or 5.3 shall incorporate the signed crit header parameter.

**Syntax**

The crit header parameter shall have the syntax specified in IETF RFC 7515 [2], clause 4.1.11.

The elements of the crit JSON array shall be the names of all the signed header parameters that are present in the JAdES signatures and specified in clause 5.2, and the string "etsiU" if the etsiU unsigned header parameter (specified in clause The etsiU header parameter5.3.1 of the present document) is present.

If the string "etsiU" appears as one of the elements of the crit JSON array all the corresponding JSON values present within the etsiU unsigned header parameter shall be treated in the same way that critical header parameters are treated in IETF RFC 7515 [2].

## New signed header parameters

### The sigT (claimed signing time) header parameter

**Semantics**

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

The sigT header parameter’s value shall specify the time at which the signer claims to have performed the signing process.

**Syntax**

The sigT header parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"sigT": {"type": "string", "format": "date-time"},

The contents of the string:

1. Shall be formatted as specified in IETF RFC 3339: "Date and Time on the Internet: Timestamps" [16].
2. Shall be the UTC time for date and time.
3. Shall not contain the part corresponding to seconds.

EXAMPLE: "2019-11-19T17:28:15Z"

### The x5t#o (X509 certificate digest) header parameter

**Semantics**

The x5t#o header parameter shall be a signed header parameter that qualifies the signature.

The x5t#o header parameter shall contain an array of references to certificates. Each reference shall contain an identifier of a digest algorithm and the digest value of the referenced certificate.

The x5t#o header parameter 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.

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.

The first reference in the array shall be a reference of the signing certificate.

The x5t#o header parameter shall not contain any other information.

**Syntax**

The x5t#o header parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"x5t#o": {

 "type": "array",

 "items": {"$ref": "#/definitions/digAlgVal"},

 "minItems": 1

},

"digAlgVal": {

 "type": "object",

 "properties": {

 "digAlg": {"type": "string", "format": "uri"},

 "digVal": {"type": "string", "contentEncoding": "base64"}

 },

 "required": ["digAlg","digVal"]

},

If header parameters x5c (specified in clause 4.1.6 of IETF RFC 7515 [2]) and x5t#S256 (specified in IETF RFC 7515 [2], clause 4.1.8) are absent, then the x5t#o header parameter shall be incorporated in the JWS Protected Header.

If the x5t#o header parameter is incorporated in the JWS Protected Header, the header parameter x5t#S256 shall be absent.

If the x5t#o header parameter is incorporated in the JWS Protected Header, then the x5c may be absent.

The digAlg member of each element of the array shall identify the digest algorithm.

The digVal member of each element of the array shall contain the base64 encoded value of the digest computed on the DER-encoded certificate.

### The srCm (signer commitment) header parameter

**Semantics**

The srCm header parameter shall be a signed header parameter that qualifies signed data object.

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

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

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

NOTE 2: 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 3: The specification of commitment type identifiers is outside the scope of the present document. For a list of predefined commitment type identifiers, see ETSI TS 119 172-1 [i.8].

**Syntax**

This header parameter shall be carried in the JWS Protected Header.

The srCm header parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"srCm":{

 "type": "object",

 "properties":{

 "commId": {"$ref": "#/definitions/oId"},

 "commQuals":{

 "type": "array",

 "items": {"type":"object"},

 }

 },

 "required": ["commId"]

},

The commId member is an instance of oId type, which is specified in clause XXX of the present document, whose id member shall have a URI as value, uniquely identifying one commitment made by the signer.

The commQuals member provides means to include additional qualifying information on the commitment made by the signer.

### The sigPl (signature production place) header parameter

**Semantics**

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

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

**Syntax**

This header parameter shall be carried in the JWS Protected Header.

The sigPl header parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"sigPlace":{

 "type": "object",

 "properties":{

 "city": {"type": "string"},

 "strAddr": {"type": "string"},

 "statProv": {"type": "string"},

 "postCode": {"type": "string"},

 "country": {"type": "string"}

 },

 "minProperties": 1

},

### The srAts (signer attributes) header parameter

**Semantics**

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

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

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

**Syntax**

This header parameter shall be carried in the JWS Protected Header.

The srAts header parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"srAts":{

 "type": "object",

 "properties": {

 "claimed": {

 "type": "array",

 "items": {"$ref": "#/definitions/claimeddAttr"},

 "minItems": 1

 },

 "certified":{

 "type": "array",

 "items": {"$ref": "#/definitions/certifiedAttr"},

 "minItems": 1

 },

 "signedAssertions": {

 "type": "array",

 "items": {"type": "string", "contentEncoding" : "base64"},

 "minItems": 1

 }

 },

 "minProperties": 1

},

"claimedAttr": {

 "type": "string",", "contentEncoding" : "base64"

},

"certifiedAttrs": {

 "oneOf":[

 {

 "x509AttrCert":{

 "type": "object",

 "properties": {

 "val": {"type": "string", "contentEncoding" : "base64"}

 },

 "required": ["val"]

 }

 },

 {

 "otherAttrCert":{"type": "pkiOb"}

 }

 ]

},

The claimed member shall contain a non-empty array of attributes claimed by the signer but which are not certified. The member val of the item shall contain the claimed attribute. The member specifiedBy shall include the name of the entity that has specified the claimed attribute. The member specRef shall contain an URI pointing to the document specifying the claimed attribute itself.

Content types for claimed attributes may be defined on a domain application basis and be part of this component.

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

* the base64 encoding of DER‑encoded X509 attribute certificates conformant to Recommendation ITU‑T X.509 [7] issued to the signer, within the X509AttrCert member; or
* attribute certificates (issued, in consequence, by Attribute Authorities) in different syntax than the one specified in Recommendation ITU‑T X.509 [7], within the OtherAttrCert member. The definition of specific OtherAttrCert is outside of the scope of the present document.

The signedAssertions member shall contain a non-empty array of base64 encoded assertions signed by a third party.

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

The definition of specific content types for signedAssertions is outside of the scope of the present document.

NOTE 2: A possible content can be a signed SAML [i.9] assertion.

Empty srAts header parameters shall not be generated.

### The adoTst (signed data time-stamp) header parameter

**Semantics**

The adoTst header parameter shall be a signed header parameter that qualifies the signed data objects.

The adoTst header parameter shall encapsulate one or more electronic time-stamps, generated before the signature production, whose message imprint computation input shall be the concatenation of the contents of all the data objects signed by the JAdES signature.

**Syntax**

This header parameter shall be carried in the JWS Protected Header.

The adoTst header parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"datObjTST": {"$ref": "#/definitions/tstContainer"},

The input of the message imprint computation for the time-stamp tokens encapsulated by adoTst signed header parameter shall be performed as indicated below:

1. Initialize the octet stream to an empty stream.
2. If JAdES signature incorporates the payload component, then concatenate its value (which is the base64url-encoded data to be signed) to the octet stream.
3. Else, if the JAdES signature incorporates the sigD header parameter specified in clause 5.2.8 of the present document, then:
* For each reference to one data object within the ordered list of references present within the aforementioned header parameter:
* Retrieve the referenced data object.
* Base64url encode the retrieved data object
* Concatenate the result to the octet stream.
1. Else, if the JAdES signature does not incorporate neither the payload component nor the sigData header parameter, then retrieve the octets of the detached data to be signed other means (which are out of the scope of the present document), base64url encode them, and concatenate the result to the octet stream.

The adoTst header parameter shall not contain the canonAlg member.

### The sigPId (signature policy identifier) header parameter

#### Semantics and syntax

**Semantics**

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

The sigPId header parameter shall contain either an explicit identifier of a signature policy or an indication that there is an implied signature policy that the relying party should be aware of.

NOTE 1: ETSI TS 119 172-1 [i.8] specifies a framework for signature policies.

**Syntax**

This header parameter shall be carried in the JWS Protected Header.

The sigPId header parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"sigPId": {

 "type": "object",

 "properties": {

 "id": {"$ref": "#/definitions/oId"},

 "hashAV": {"$ref": "#/definitions/digAlgVal"},

 "hashPSp": {"type": "boolean"},

 "sigPQuals": {

 "type": "array",

 "minItems": 1

 }

 },

 "required": ["id"]

},

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

The hashAV member shall contain the identifier of the digest algorithm and the digest value of the object obtained after processing id.

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

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

The sigPQuals member may contain one or more qualifiers of the same type.

Clause 5.2.7.2 specifies three signature policy qualifiers.

#### Signature policy qualifiers

**Semantics**

This clause specifies the following three qualifiers for the signature policy:

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

**Syntax**

The spURI , spUserNotice , and spDSpec qualifiers shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and are copied below for information.

"spURI": {"type": "string", "format": "uri"},

"spUserNotice": {

 "type": "object",

 "properties": {

 "noticeRef": {

 "type": "object",

 "properties": {

 "organization": {"type": "string"},

 "noticeNumbers": {

 "type": "array",

 "items": {"type": "integer"},

 "minItems" : 1

 }

 }

 },

 "explText": {"type": "string"}

 },

 "minProperties": 1

},

"spDSpec": {"$ref": "#/definitions/oId"},

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

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

The spUserNotice qualifier shall contain information that is intended for being displayed whenever the signature is validated.

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

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

### The sigD header parameter

#### Semantics and Syntax

**Semantics**

The sigD header parameter shall be a signed header parameter.

The sigD header parameter shall not appear in JAdES signatures that include the JWS Payload.

The sigD header parameter may appear in JAdES signatures whose signed data objects are detached.

A JAdES signature shall have at most one sigD header parameter.

The sigD header parameter shall:

1. Allow to reference one or more data objects that are signed by the JAdES signature.
2. Reference the signed data objects with a set of parameters.
3. Not support chaining of references shall not be allowed. Only the data objects directly referenced within the sigD header parameter shall be signed. If some referenced object contains in its turn references to other objects, these last objects shall not be signed.

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

1. Allow to specify how the aforementioned data objects shall contribute to the sequence of octets that shall be the input for the computation of the JWS Signature Value.
2. Allow to define different mechanisms for meeting the two aforementioned requirements.

The sigD header parameter may also incorporate base64-encoded digest values of the referenced data objects within one string.

The sigD header parameter may also incorporate any additional information for meeting requirements 1), 2), and 4) as required by the mechanisms mentioned in 5).

**Syntax**

This header parameter shall be carried in the JWS Protected Header.

The sigD header parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"sigD":{

 "type": "object",

 "properties": {

 "mId" : {"type":"string", "format": "uri"},

 "pars" : {

 "type": "array",

 "items": {"type": "string"}

 },

 "hashm" : {"type":"string"},

 "hashv" : {

 "type": "array",

 "items": {"type": "string"}

 },

 "ctys" : {

 "type": "array",

 "items": {"type": "string"}

 },

 "minItems": 1

 },

 "required" : ["mId"]

}

The mId member shall be an URI identifying the mechanism used for referencing each signed data object. The present document defines 3 referencing mechanisms with their corresponding identifiers in clauses 5.2.8.2, 5.2.8.3, and 5.2.8.4.

The pars member shall be an array of strings. Each element of the array shall contain a reference to one signed data object, as required by the identification mechanism identified in the mId member.

The hashm member shall be a string identifying a digest algorithm.

The hashv member shall be an array of strings. Each element of the array shall contain the base64-encoded digest value of the signed data object referenced by the parameter value that is present in the same position of the pars array. The presence of this member is conditional on the definition of the identification mechanism. If this member is present, then hashm member shall be present.

The ctys member shall contain an array of strings. The contents of each component of this array shall have the same semantics of the cty header parameter specified in clause 4.1.10 of IETF RFC 7515 [2].

There shall be as many elements within the ctys aray as signed data objects. Each component of this array shall contain the information corresponding to the signed data object referenced by the parameter value that is present in the same position of the pars array, except if the content type is implied by the signed data object or the signed data object is a counter-signed signature: in these cases, the element of the ctys array shall have as value an empty string.

#### Mechanism HttpHeaders

The URL identifying this referencing mechanism shall be "http: //uri.etsi.org/19182/HttpHeaders".

For this referencing mechanism, neither hashv, neither hashm member, nor ctys shall be present.

Using this referencing mechanism, a JAdES signature may explicitly reference several HTTP headers and sign them, as well as the HTTP message body.

For this referencing mechanism, the contents of the pars member shall be an array of lowercased names of HTTP header fields, each one with the semantics and syntax specified in clause 2.1.3 of draft-cavage-http-signatures-10: "Signing HTTP Messages" [17].

The HTTP headers whose names appear in the pars member shall be the names of the HTTP headers that shall be concatenated to form the input octets to the computation of JWS Signature Value as specified in clause 2.3 of draft-cavage-http-signatures-10 [17].

NOTE: As specified in clause 3.1 of draft-cavage-http-signatures-10 [17], the HTTP message body can also be signed by incorporating into the HTTP message the "Digest" HTTP header specified in clause 4.3.2 of IETF RFC 3230: "Instance Digests in HTTP" [18].

ADVICE REQUEST BY EDITOR: draft-cavage-http-signatures-10 [17] is a draft document, which is used in banking environments. How to deal with the fact that from a TS we are referencing something that it is a draft?

#### Mechanism ObjectIdByURI

The URL identifying this referencing mechanism shall be "http: //uri.etsi.org/19182/ObjectIdByURI".

For this referencing mechanism, neither hashv, nor hashm, nor any other member shall be present. Member ctys may be present.

For this referencing mechanisms, the contents of the pars member shall be an array of strings. Each string shall be an URI referencing one data object.

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

The stream of octets corresponding to the contribution of the signed data to the computation of the JWS Signature Value shall be generated as indicated below:

1. Initialize the stream of octets to an empty stream.
2. While there are URIs in the pars array not visited,
* Take the next one.
* Dereference the URI.
* Concatenate the resulting octets to the stream of octets to be signed.
1. If the b64 header parameter specified in clause 3 of IETF RFC 7797 [15] is absent or is present and set to "true", then base64url encode the resulting stream of octets.

#### Mechanism ObjectIdByURIHash

The URL identifying this referencing mechanism shall be "http: //uri.etsi.org/19182/ObjectIdByURIHash".

For this referencing mechanism, the hashv, and the hashm members shall be present. Member ctys may be present.

For this referencing mechanisms, the contents of the pars member shall be an array of strings. Each string shall be an URI referencing one signed data object.

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

NOTE 1: As this sigD is a signed header parameter, and it already includes the digest of the referenced data objects that are signed by the JAdES signature, they are indirectly signed by signing the sigD signed header parameter.

## New unsigned header parameter

### The etsiU header parameter

**Semantics**

The etsiU unprotected header parameter shall be a JSON array whose elements contain JSON values that are not signed by the JAdES signature.

NOTE 1: The rationale for this is as follows: the computation of certain time-stamp tokens message imprints is performed by digesting the concatenation of sets of unsigned header parameters, and this concatenation needs to be performed following an order; the JSON array allows to define such an order: the unsigned header parameters are concatenated following the order of appearance within the JSON array.

NOTE 2: As it has been specified in clause 4 of the present document etsiU header parameter is incorporated in the JWS Unprotected Header specified in clause 3.2 of IETF RFC 7515[2]. Consequently, all its elements will also be unprotected, and its elements will play in JAdES signatures the same role as the role played by the unsigned attributes for CAdES signatures, and the role played by the unsigned qualifying properties for XAdES signature.

The etsiU header parameter shall contain JSON values that qualify the JAdES signature itself, or the signer, or some of the signed data objects.

All the unprotected JSON values in JAdES signatures shall be placed as components of the etsiU container.

NOTE 2: If the etsiU header parameter is present then JWS JSON Serialization as specified in RFC 7515 [?] section 3.2, needs to be employed as the alternative JWS Compact Serialization does not support unprotected header.

The header parameters present within the etsiU header parameter shall appear as clear instances of unsigned header parameters or as their corresponding base64url encodings.

NOTE 3: While clear instances of unsigned header parameters require some type of canonicalization if they contribute to the computation of a time-stamp message imprint, their base64-encoded values will not require such canonicalization. The present document is neutral about which alternative should be used.

The present document specifies:

1. A JSON object (sigPSt) containing details for facilitating access to a signature policy document, in clause 5.3.3.
2. A JSON object (cSig) containing details for containing a counter-signature of the JAdES signature itself, in clause 5.3.2.
3. A JSON object (sigTst) containing a time-stamp token on the JWS Signature Value, in clause 5.3.4.
4. A JSON object (xVals) containing CA certificates required for validating the signature, in clause 5.3.5.1.
5. A JSON object (rVals) containing values of revocation data required for validating the signature, in clause 5.3.5.2.
6. A JSON object (axVals) containing certificates of Attribute Authorities required for validating the signature, in clause 5.3.5.3.
7. A JSON object (arVals) containing values of revocation data of Attribute Authorities required for validating the signature, in clause 5.3.5.4.
8. A JSON object (tstVd) containing validation data (certificate and values of revocation data) for time-stamp tokens present in the signature, in clause 5.3.6.1.
9. A JSON object (arcTst) containing one or more time-stamp tokens on all the data objects incorporated into the JAdES signature, in clause 5.3.6.2.
10. A JSON object (xRefs) containing references to certificates required for validating the signature, in clause A.1.1.
11. A JSON object (rRefs) containing references to revocation data required for validating the signature, in clause A.1.2.
12. A JSON object (axRefs) containing references to certificates of Attribute Authorities required for validating the signature, in clause A.1.3.
13. A JSON object (arRefs) containing references to revocation data of Attribute Authorities required for validating the signature, in clause A.1.4.
14. A JSON object (sigRTst) containing a time-stamp token on the references to the validation material and the JWS Signature Value, in clause A.1.5.1.
15. A JSON object (rfsTst) containing a time-stamp token on the references to the validation material, in clause A.1.5.2.

**Syntax**

The etsiU header parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"etsi": {

 "type": "array",

 "minItems": 1

},

The etsiU header parameter shall be a non-empty array.

The etsiU header parameter shall be incorporated as member of the header JSON object of the JSON Web Signature.

NOTE 1: The header component is the place reserved by IETF RFC 7515 [2] for unsigned header parameters in JWS signatures. Clause 3.2 of IETF RFC 7515 [2] leaves its content open. The present document suitably profiles its contents.

The content of any element of the etsiU array shall be either an unsigned JSON value in clear (clear JSON incorporation), or its base64url encoding (base64url incorporation).

The array shall not contain JSON values in clear in some positions, and base64url encoded unsigned JSON values in others. Either all of them shall be incorporated in clear or shall be incorporated base64url encoded.

The etsiU header parameter should be the only header parameter incorporated to the JWS Unprotected Header. Any unprotected JSON value that is not specified in the present document should be incorporated as an element of the etsiU header parameter.

NOTE 2: Adding these header parameters into the etsiU header parameter allows to properly secure them in the long-term using the arcTst unsigned header parameter.

If the etsiU header parameter contains JSON values in clear, instances of tstContainer type shall have the canonAlg member.

If the etsiU header parameter contains bas64url-encoded JSON values, instances of tstContainer type shall not have the canonAlg member.

### The cSig (counter signature) JSON object

**Semantics**

The cSig JSON object shall contain one counter signature of the JAdES signature where cSig is incorporated. This counter signature may also be a JAdES signature.

**Syntax**

This JSON object shall be carried in the JWS Unprotected Header.

The cSig JSON object contains either a JSON Web Signature or a JAdES signature that signs the JWS Signature Value of the embedding JAdES signature.

One JAdES-WS counter signature may itself be counter signed using a cSig JSON object, signing the JWS Signature Value of the first counter signature, built as described above.

NOTE 1: This is an alternative way of constructing arbitrarily long series of counter signatures, each one signing the JWS Signature Value of the one where it is directly embedded.

### The sigPSt JSON object

**Semantics**

The sigPSt JSON object shall contain either:

* the signature policy document which is referenced in the sigPId JSON object 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**

This JSON object shall be carried in the JWS Unprotected Header.

The sigPSt shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and are copied below for information.

"sigPolStore": {

 "oneOf": [

 {

 "type": "object",

 "properties": {

 "sigPolDoc": {"type": "string", "contentEncoding": "base64"}

 }

 },

 {

 "type": "object",

 "properties": {

 "sigPolLocalURI": {"type": "string", "format": "uri-reference"}

 }

 }

 ],

 "type":"object",

 "properties":{

 "spDSpec": {"$ref": "#/definitions/oId"}

 },

 "minProperties": 1

},

The sigPolDoc member shall contain the base64 encoded signature policy.

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

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

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

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

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

### The sigTst JSON object

**Semantics**

The sigTst shall be a member of etsiU qualifying the signature.

The sigTst JSON object shall encapsulate one or more electronic time-stamps time-stamping the JWS Signature Value component.

**Syntax**

This JSON object shall be carried in the JWS Unprotected Header.

The sigTst JSON object shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"SigTst": {"$ref": "#/definitions/tstContainer"},

The input of the message imprint computation for the time-stamp tokens encapsulated by sigTst JSON object shall be the content of the signature component (which is the base64url encoded JWS Signature Value) of the JAdES signature.

The sigTst JSON object shall not contain the canonAlg member.

### JSON objects for validation data values

#### The xVals JSON array

**Semantics**

The xVals JSON array:

1. Shall contain the certificate of the trust anchor, if such certificate does exist and if it is not already present within other component of the underlying JSON Web Signature. If this certificate is present within another component of the underlying JSON Web Signature, it should not be included.
2. Shall contain the CA certificates within the signing certificate path that are not already present within other component of the underlying JSON Web Signature. The certificates present within other component of the underlying JSON Web Signature should not be included.
3. Shall contain the signing certificate if it is not already present within other component of the underlying JSON Web Signature. If this certificate is present within other component of the underlying JSON Web Signature, it should not be included.
4. Shall contain certificates used to sign revocation status information (e.g. CRLs or OCSP responses) of certificates in 1), 2), and 3), and certificates within their respective certificate paths that are not present in the signature. Certificate values present within the signature, including certificate values within the revocation status information themselves should not be included.
5. Shall not contain CA certificates that pertain exclusively to the certificate paths of certificates used to sign attribute certificates or signed assertions within srAts, or electronic time-stamps. And
6. May contain a set of certificates used to validate any cSignature incorporated into the JAdES signature that are not present in other components of the JAdES signature or its cSignatures. This set may include any of the certificates listed in 1), 2), 3) and 4) referred to signing certificates of cSignatures instead of the signing certificate of the JAdES signature. The certificates present elsewhere in the JAdES signature or its cSignatures should not be included.

**Syntax**

The xVals array parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"cVals": {

 "type": "array",

 "items": {

 "oneOf": [

 {"x509Cert": {"$ref": "#/definitions/pkiOb"}},

 {"otherCert": {"$ref": "#/definitions/pkiOb"}}

 ]

 },

 "minItems": 1

},

An x509Cert item shall contain the base64 encoding of a DER‑encoded X.509 certificate.

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

#### The rVals JSON object

**Semantics**

The rVals JSON object:

1. Shall contain revocation values corresponding to CA certificates within the signing certificate path if they are not already present within other component of the underlying JSON Web Signature. It shall not contain a revocation value for the trust anchor. The revocation values present within other component of the underlying JSON Web Signature should not be included.
2. Shall contain a revocation value for the signing certificate if it is not already present within other component of the underlying JSON Web Signature. If it is present within other component of the underlying JSON it should not be included.
3. May contain revocation values corresponding to certificates used to sign CRLs or OCSP responses of 1) and 2), and certificates within their respective certificate paths. The revocation values present already present within other component of the underlying JSON Web Signature should not be included.
4. Shall not contain revocation values corresponding to CA certificates that pertain exclusively to the certificate paths of certificates used to sign attribute certificates or signed assertions within srAts, or electronic time-stamps. And
5. May contain revocation values corresponding to the signing certificate of any cSignature incorporated into the JAdES signature as well as to the CA certificates in its certificate path. This set may include any of the revocation values listed in 1), 2), and 3) referred to signing certificates of cSignatures instead of the signing certificate of the JAdES signature. However, those revocation values among the aforementioned ones that are already present in other components of the JAdES signature should not be included.

**Syntax**

This JSON object shall be carried in the JWS Unprotected Header.

The rVals JSON object shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"rVals": {

 "type": "object",

 "properties":{

 "crlVals": {

 "type": "array",

 "items": {"$ref":"#/definitions/pkiOb"},

 "minItems": 1

 },

 "ocspVals": {

 "type": "array",

 "items": {"$ref":"#/definitions/pkiOb"},

 "minItems": 1

 },

 "otherVals": {

 "type": "array",

 "items": {"type":"object"},

 "minItems": 1

 }

 },

 "minProperties": 1

},

crlVals member shall be a non-empty array of encoded X.509 CRLs [9].

Each item of crlVals array shall contain the base64 encoding of a DER‑encoded X.509 CRLs [9].

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

ocspVals member shall be a non-empty array of encoded OCSP responses [10].

Each item of ocspVals array shall contain the base64 encoding of a DER‑encoded OCSPResponse defined in IETF RFC 6960 [10].

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

#### The axVals JSON array

**Semantics**

The axVals JSON array:

1. shall contain the value(s) of the signing certificate(s) of the attribute certificate(s) and signed assertion(s) incorporated into the XAdES signature;
2. shall contain, if not present within the signature, the value(s) of the certificate(s) for the trust anchor(s) if such certificates exist, and the CA certificate values within path of the signing certificate(s) of the attribute certificate(s) and signed assertion(s) incorporated into the XAdES signature. Certificate values present within the signature should not be included; and
3. may contain the certificate values used to sign CRLs or OCSP responses and the certificates values within their respective certificate paths, used for validating the signing certificate(s) of the attribute certificate(s) and signed assertion(s) incorporated into the XAdES signature. Certificate values present within the signature, including certificate values within the revocation status information themselves should not be included.

**Syntax**

The axVals JSON array shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"acVals": {"$ref": "#/definitions/cVals"}

#### The arVals JSON object

**Semantics**

The arVals JSON object:

1. shall contain the revocation value(s) of the certificate(s) that sign the attribute certificate(s) and signed assertion(s) incorporated into the XAdES signature;
2. shall contain, if not incorporated into the signature, 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 into the XAdES signature. It shall not contain revocation values for the trust anchors. Values already incorporated into the signature should not be included; and
3. may contain 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 into the XAdES signature. Revocation values already incorporated into the signature should not be included.

**Syntax**

The arVals header parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"arVals": {"$ref": "#/definitions/rVals"}

If the validation data contain one or more Delta CRLs, this header parameter shall include the set of CRLs required to provide complete revocation lists.

### JSON values for long term availability and integrity of validation material

#### The tstVd JSON object

**Semantics**

The tstVd JSON object shall be a container for validation data required for carrying a full verification of the electronic time-stamps embedded within any of the different electronic time-stamp container JSON objects defined in the present document.

The tstVd JSON object shall allow incorporating certificate values.

The tstVd JSON object shall allow incorporating revocation values.

**Syntax**

The tstVd JSON object shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"tstVD": {

 "type": "object",

 "properties": {

 "certVals": {"$ref": "#/definitions/cVals"},

 "revVals": {"$ref": "#/definitions/rVals"},

 "onSdo": {"type": "boolean"}

 },

 "minProperties": 1

},

The certVals member shall contain certificates used in the full verification of electronic time-stamps.

The certVals member may contain all the certificates required for a full verification of the electronic time-stamps.

The certVals member may also contain only some of the certificate values if the rest are present elsewhere in the JAdES signature (for instance within the electronic time-stamp itself, or in other tstVd created for other electronic time-stamps).

The revVals member shall contain revocation values used in the full verification of electronic time-stamps embedded in one JAdES time-stamp container.

The revVals member may contain all the revocation values required for a full verification of the electronic time-stamps.

The revVals member may also contain only some of the revocation values if the rest are present elsewhere in the JAdES signature (for instance within the electronic time-stamp itself, or in other tstVd created for other electronic time-stamps).

The onSdo member is a boolean value notifying whether the electronic time-stamp whose validation material contains the tstVd JSON object, time-stamps the signed data objects or not, (i.e. whether the aforementioned electronic time-stamp is enclosed within the adoTst header parameter or not).

If the value of the onSdo member is set to "true", then it shall indicate that the electronic time-stamp whose validation material contains the tstVd JSON object, time-stamps the signed data objects.

If the onSdo member is absent or it is present and its value is set to "false", then it shall indicate that the electronic time-stamp whose validation material contains the tstVd JSON object, does not time-stamp the signed data objects.

If The tstVd JSON object contains validation data for time-stamp tokens encapsulated in the adoTst header parameter then:

1. It shall be added at the end of the array within the etsiU header parameter.
2. It shall include the onSdo member set to value "true".

If the tstVd JSON object contains validation data for time-stamp tokens that are encapsulated in a JSON object different than the adoTst header parameter, then:

1. It shall be added in the array of the etsiU header parameter immediately after the item containing the aforementioned JSON object containing the electronic time-stamp.
2. It shall not include the onSdo member.

#### The arcTst JSON object

**Semantics**

The arcTst JSON object shall encapsulate electronic time-stamps computed on all the data objects incorporated into the JAdES signature at the time of generating each electronic time-stamp.

NOTE 1: The purpose of this JSON object is to tackle the long term availability and integrity of the validation material.

**Syntax**

The arcTst JSON object shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"arcTst": {

 "type": "object",

 "properties": {

 "tstContainer": {"$ref": "#/definitions/tstContainer"},

 "timeStamped": {

 "type": "string",

 "enum": ["all", "previousArcTst"]

 }

 }

},

If the JAdES signature incorporates a cSig JSON object, all the required material for conducting the validation of the counter-signature shall be incorporated into the JAdES signature before generating the first arcTst JSON object. This may be done within the counter-signature itself or within the containers available within the counter-signed JAdES signature.

The contents of the cSig JSON object should not be changed, once it has been time‑stamped by the arcTst.

NOTE 2: If a cSig JSON object is time‑stamped by the arcTst, any ulterior change of their contents (by addition of unsigned JSON values if the counter‑signature is a JAdES signature, for instance) would make the validation of the arcTst and, in consequence, the validation of the cSigned JAdES signature, fail.

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

The timeStamped member shall have two possible values, namely: "all" and "previousArcTst".

#### Computation of message-imprint

##### Time-stamping the time-stamp token of the last arcTst

If the value of timeStamped is equal to "previousArcTst" the time-stamp tokens within the container shall time-stamp the last existing arcTst container in the JAdES signature and its associated tstVD, if it is required to generate and incorporate it into the JAdES signature. In consequence the message imprint computation input shall be either the last existing arcTst container, or the concatenation of this container and its associated tstVD, either canonicalized or not.

Absence of timeStamped shall be treated as if it is present with value "all".

##### Time-stamping all the contents of the JAdES signature

###### Base64url incorporation

If the value of timeStamped is equal to "all" or it is absent, and the etsiU array contains base64url encoded unsigned JSON values, then the message imprint computation input shall be the concatenation of the components in the order they are listed below:

1. One of the following:
* The value of payload member, if present.
* The base64url encoded stream of octets retrieved after processing the sigD header parameter if present.
* The base64url encoded stream of octets of the detached payload retrieved by other means, (out of the scope of the present document) if both the payload component sigD header parameter are absent.
1. The character '.'.
2. The value of protected member, which is also base64url encoded, followed by the character '.'.
3. The value of the signature member, which is the base64url encoded JWS Signature Value.
4. The result of taking the contents of the etsiU array in the order they appear within the array, and concatenating them to the final octet stream. While concatenating, the following rules apply:

a) the xVals JSON array shall be incorporated, base64url encoded, into the signature if it is not already present and the signature misses some of the certificates listed in clause 5.3.5.1 that are required to validate the JAdES signature;

b) the rVals JSON object shall be incorporated, base64url encoded, into the signature if it is not already present and the signature misses some of the revocation data listed in clause 5.3.5.2 that are required to validate the JAdES signature;

c) the axVals JSON array shall be incorporated, base64url encoded, into the signature if not already present and the following conditions are true: attribute certificate(s) or signed assertions have been incorporated into the signature, and the signature misses some certificates required for their validation; and

d) the arVals JSON object shall be incorporated, base64url encoded, into the signature if not already present and the following conditions are true: attribute certificates or signed assertions have been incorporated into the signature, and the signature misses some revocation values required for their validation.

###### Clear JSON incorporation

If the value of timeStamped is equal to "all" or it is absent, and the and the etsiU array contains unsigned header parameters in clear, then the message imprint computation input shall be the concatenation of the components in the order they are listed below:

1. One of the following:
* The value of payload member, if present.
* The base64url encoded stream of octets retrieved after processing the sigData header parameter if present.
* The base64url encoded stream of octets of the detached payload retrieved by other means, (out of the scope of the present document) if both the payload member sigData header parameter are absent.
1. The character '.'.
2. The value of protected member, which is also base64url encoded, followed by the character '.'.
3. The value of the signature member, which is the base64url encoded JWS Signature Value.
4. The result of taking the contents of the etsiU array in order, canonicalizing each one of them using the canonicalization algorithm identified in canonAlg member, and concatenating each resulting octet stream to the final octet stream. While concatenating, the following rules apply:

a) the xVals JSON array shall be canonicalized and incorporated, in clear JSON, into the signature if it is not already present and the signature misses some of the certificates listed in clause 5.3.5.1 that are required to validate the JAdES signature;

b) the rVals JSON object shall be canonicalized and incorporated, in clear JSON, into the signature if it is not already present and the signature misses some of the revocation data listed in clause 5.3.5.2 that are required to validate the JAdES signature;

c) the axVals JSON array shall be canonicalized and incorporated, in clear JSON, into the signature if not already present and the following conditions are true: attribute certificate(s) or signed assertions have been incorporated into the signature, and the signature misses some certificates required for their validation; and

d) the arVals JSON object shall be canonicalized and incorporated, in clear JSON, into the signature if not already present and the following conditions are true: attribute certificates or signed assertions have been incorporated into the signature, and the signature misses some revocation values required for their validation.

### The sigPSt JSON object

**Semantics**

The sigPSt JSON object shall contain either:

* the signature policy document which is referenced in the sigPId header parameter 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 sigPSt JSON object shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and are copied below for information.

"sigPolStore": {

 "oneOf": [

 {

 "type": "object",

 "properties": {

 "sigPolDoc": {"type": "string", "contentEncoding": "base64"}

 }

 },

 {

 "type": "object",

 "properties": {

 "sigPolLocalURI": {"type": "string", "format": "uri-reference"}

 }

 }

 ],

 "type":"object",

 "properties":{

 "spDSpec": {"$ref": "#/definitions/oId"}

 },

 "minProperties": 1

},

The sigPolDoc member shall contain the base64 encoded signature policy.

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

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

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

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

NOTE 3: Being an unsigned JSON object, it is not protected by the digital signature. If the sigPId header parameter is incorporated into the signature and contains the hashAV member with the digest value of the signature policy document, any alteration of the signature policy document present within sigPSt or within a local store, would be detected by the failure of the digests comparison.

## Generally useful syntax

### The oId data type

**Semantics**

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

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

Instances of oId data type may contain a number of references to documents where additional information about the nature of the data object qualified by the instance of the bjectId data type, can be found.

**Syntax**

The oId shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"oId": {

 "type": "object",

 "properties": {

 "id": {"type": "string", "format": "uri"},

 "desc": {"type": "string"},

 "docRefs":{

 "type": "array",

 "items": {"type": "string", "format": "uri"},

 "minItems": 1

 }

 },

 "required": ["id"]

},

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

The value of the id member shall be an URI. If the identifier of the object is an OID then the value of this member shall be encoded as an URN as specified by the IETF RFC 3061: "A URN Namespace of Object Identifiers" [4].

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

The desc member shall contain an informal text describing the object.

The docRefs member shall contain an arbitrary number of URI values pointing to further explanatory documentation of the data object.

### The pkiOb data type

**Semantics**

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

NOTE 1: Examples of such PKI objects, include X.509 certificates and revocation lists, OCSP responses, attribute certificates, and electronic time-stamps.

**Syntax**

The pkiOb type shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"pkiOb": {

 "type": "object",

 "properties":{

 "encoding": {"type": "string", "format": "uri"},

 "specRef": {"type": "string"},

 "val": {"type": "string", "contentEncoding" : "base64"}

 },

 "required": ["val"]

},

The content of this data type shall be the PKI object, base64 encoded.

EDITOR’S REQUEST OF ADVICE: SHOULD WE GET RID OF ENCODING AND SPECREF, AND FORCE ALWAYS DER?. ARE WE SURE THAT ANY OBJECT SHALL BE DER-ENCODED?

The encoding member's value shall be a URI identifying the encoding used in the original PKI object. The values for the URI shall be one of the values defined in clause 5.1.3 of ETSI EN 319 132-1: "Electronic Signatures and Infrastructures (ESI); XAdES digital signatures; Part 1: Building blocks and XAdES baseline signatures" [5].

If the encoding member is not present, then the contents of val member shall be the result of base64 encoding the DER-encoded ASN.1 data.

### Container for electronic time‑stamps

#### Introduction

The present document specifies JSON objects that act as electronic time-stamps containers.

Electronic time-stamps within the aforementioned containers may time-stamp isolated components or concatenations of several components of JAdES signatures, and/or signed data objects.

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

#### Containers for electronic time-stamps

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

* Containers for electronic time-stamps proving that the signed data object(s) has (have) been created before certain time instant: adoTst
* Container for electronic time-stamps proving that the signature value has been computed before a certain time instant (to protect against repudiation in case of a key compromise): sigTst.
* Container for electronic time-stamps time-stamping the signature and validation data values, for providing long term JAdES signatures: arcTst.
* Containers for electronic time-stamps on header parameters that contain references to validation data, namely: rfsTst and sigRTst. (specified in clause A.1.5 of the present document).

#### The tstContainer type

**Semantics**

The tstContainer type shall:

* allow encapsulating IETF RFC 3161 [8] electronic time-stamps as well as electronic time-stamps in other formats;
* allow encapsulating more than one electronic time-stamp generated for the same set of data objects (each one issued by different TSAs, for instance), and
* provide means for managing electronic time-stamps computed on a concatenation of JAdES components, and/or signed data objects (including detached).

**Syntax**

The tstContainer type shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"tstContainer": {

 "type": "object",

 "properties": {

 "canonAlg": {"type": "string", "format": "uri"},

 "tstokens": {

 "type": "array",

 items": {"$ref": "#/definitions/tstToken"}

 }

 },

 "required": ["tstokens"]

},

"tstToken":{

 "type": "object",

 "properties":{

 "type": {"type": "string"},

 "encoding": {"type": "string", "format": "uri"},

 "specRef": {"type": "string"},

 "val": {"type": "string", "contentEncoding" : "base64"}

 },

 "required": ["val"]

},

The tstContainer‘s tstokens member shall contain a non-empty array of JSON objects each one encapsulating one electronic time-stamp token.

The tsToken‘s type member shall identify the type of the time-stamp token. For RFC 3161 [8] time-stamp tokens this member shall not be present.

The tstoken‘s encoding member shall be an URI and shall identify the encoding used for the time-stamp token. For RFC 3161 [8] time-stamp tokens this member shall not be present.

The tstoken‘s specRef member shall identify the technical specification that has defined the used time-stamp token. For RFC 3161 [8] time-stamp tokens this member shall not be present.

Finally the tstoken‘s val member shall contain the base64 encoding of the electronic time-stamp token itself. For RFC 3161 [8] time-stamp tokens this member shall contain the base64 encoding of the DER-encoded electronic time-stamp token.

When building JAdES, and unsigned header parameters are time-stamped, each time-stamp container implicitly identifies what unsigned header parameters are time-stamped by the electronic time-stamps and how they contribute to the input of the message imprint's computation. No further information in the time-stamp token container is required.

NOTE 1: This is because all the components of a JAdES signature are placed within JAdES signature itself.

NOTE 2: time-stamp tokens in JAdES signatures usually time-stamp unsigned header parameters and/or other components.

The tstContainer’s canonAlg member shall contain the identifier of a canonicalization algorithm.

If the tstContainer’s canonAlg member is present, then the bytes concatenated for building the time-stamp’s message imprint input, shall be the bytes resulting from applying the canonicalization algorithm to all the time-stamped data objects.

If the tstContainer’s canonAlg is absent then the bytes concatenated for building the time-stamp’s message imprint input, shall be the bytes of each of the time-stamped data object themselves.

# JAdES baseline signatures

## Signature levels

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

1. B-B level provides requirements for the incorporation of signed and some unsigned header parameters when the signature is generated.
2. B-T level provides requirements for the generation and inclusion, for an existing signature, of a trusted token proving that the signature itself actually existed at a certain date and time.
3. 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.
4. B-LTA level provides requirements for the incorporation of electronic time-stamps that allow validation of the signature long time after its generation. This level aims to tackle the long term availability and integrity of the validation material.

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

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

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

NOTE 4: Conformance to B-LT level, when combined with appropriate additional preservation techniques tackling the long term availability and integrity of the validation material is sufficient to allow validation of the signature long time after its generation. The assessment of the effectiveness of preservation techniques for signed data other than implementing the B-LTA level are out of the scope of the present document. The reader is advised to consider legal instruments in force and/or other standards (for example ETSI TS 101 533-1 [i.10] or IETF RFC 4998 [i.11]) that can indicate other preservation techniques. Annex C defines what needs to be taken into account when using other techniques for long term availability and integrity of validation data and incorporating a new unsigned header parameter derived from these techniques into the signature.

## General requirements

### Algorithm requirements

The algorithms and key lengths used to generate and augment digital signatures should be as specified in ETSI TS 119 312: "Electronic Signatures and Infrastructures (ESI); Cryptographic Suites" [i.12].

NOTE 1: Cryptographic suites recommendations defined in ETSI TS 119 312 [i.12] can be superseded by national recommendations.

NOTE 2: IETF RFC 6931: "Additional XML Security Uniform Resource Identifiers (URIs)" [i.13] defines a set of additional XML security URIs, which complement those ones defined in W3C Recommendation: "XML Signature Syntax and Processing. Version 1.1" [i.7].

In addition, MD5 algorithm shallnot be used as digest algorithm.

### Notation for requirements

The present clause describes the notation used for defining the requirements of the different JAdES signature levels.

The requirements on the header parameters and certain other signature's components for each JAdES signature level are expressed in table 2. A row in the table either specifies requirements for a header parameter, other signature's component, or a service.

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

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

1. Column "Header parameters/Elements in etsiU unsigned header parameter/Services":

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

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

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

1. Column "Presence in B-B-Level": This cell contains the specification of the presence of the header parameter or other signature's component, or the provision of a service, for JAdES-B-B signatures.
2. Column "Presence in B-T level": This cell contains the specification of the presence of the header parameter or other signature's component, or the provision of a service, for JAdES-B-T signatures.
3. Column "Presence in B-LT level": This cell contains the specification of the presence of the header parameter or other signature's component, or the provision of a service, for JAdES-B-LT signatures.
4. Column "Presence in B-LTA level": This cell contains the specification of the presence of the header parameter or other signature's component, or the provision of a service, for JAdES-B-LTA signatures. Below follow the values that can appear in columns "Presence in B-B", "Presence in B-T", "Presence in B-LT", and "Presence in B-LTA":
* "shall be present": means that the header parameter or signature's component shall be incorporated to the signature, and shall be as specified in the document referenced in column "References", further profiled with the additional requirements referenced in column "Requirements", and with the cardinality indicated in column "Cardinality".
* "shall not be present": means that the header parameter or signature's component shall not be incorporated to the signature.
* "may be present": means that the header parameter or signature's component may be incorporated to the signature, and shall be as specified in the document referenced in column "References", further profiled with the additional requirements referenced in column "Requirements", and with the cardinality indicated in column "Cardinality".
* "shall be provided": means that the service identified in the first column of the row shall be provided as further specified in the SPO-related rows. This value only appears in rows that contain requirements for services. It does not appear in rows that contain requirements for header parameters or signature's components.
* "conditioned presence": means that the incorporation to the signature of the item identified in the first column is conditioned as per the requirements referenced in column "Requirements" and requirements in specifications and clauses referenced by column "References", with the cardinality indicated in column "Cardinality".
* "\*": means that the header parameter or signature's component (service) identified in the first column should not be incorporated to the signature (provided) in the corresponding level. Upper signature levels may specify other requirements.

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

1. Column "Cardinality": This cell indicates the cardinality of the header parameter or other signature's component. If the cardinality is the same for all the levels, only the values listed below appear. Otherwise the content specifies the cardinality for each level. See the example at the end of the present clause showing this situation. Below follows the values indicating the cardinality:
* **0:** The signature shall not incorporate any instance of the header parameter or the signature's component.
* **1:** The signature shall incorporate exactly one instance of the header parameter or the signature's component.
* **0 or 1:** The signature shall incorporate zero or one instance of the header parameter or the signature's component.
* **≥ 0:** The signature shall incorporate zero or more instances of the header parameter or the signature's component.
* **≥ 1:** The signature shall incorporate one or more instances of the header parameter or the signature's component.
1. Column "References": This shall contain either the number of the clause specifying the header parameter in the present document, or a reference to the document and clause that specifies the other signature's component.
2. Column "Additional notes and requirements": This cell contains numbers referencing notes and/or letters referencing additional requirements on the header parameter or the other signature's component. Both notes and additional requirements are listed below the table.

## Requirements on JAdES components and services

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

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

NOTE 1: JAdES-B-B signatures that incorporate only the header parameters and other components that are mandatory in table 2, and that implement the mandatory requirements, contain the lowest number of header parameters and other components, with the consequent benefits for interoperability.

In JAdES baseline signatures the header parameters that act as electronic time-stamps containers shall encapsulate only IETF RFC 3161 [8] updated by IETF RFC 5816 [11] time-stamp tokens.

Any header parameter specified in IETF RFC 7515[2] or IETF RFC 7797[15], and not further profiled in clause 5.1, may be present (cardinality of 0 or 1) in the four levels defined in Table 1 below.

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

| Header parameters/Elements in etsiU unsigned header parameter/Services | Presence inB-B level | Presence inB-T level | Presence inB-LT level | Presence inB-LTA level | Cardinality | References | Additional requirements and notes |
| --- | --- | --- | --- | --- | --- | --- | --- |
| alg | shall be present | shall be present | shall be present | shall be present | 1 | Clause 5.1.2 |  |
| cty | conditioned presence | conditioned presence | conditioned presence | conditioned presence | 0 or 1 | Clause 5.1.3 | 2 |
| kid  | may be present | may be present | may be present | may be present | 0 or 1 | Clause 5.1.4 |  |
| x5u | may be present | may be present | may be present | may be present | 0 or 1 | Clause 5.1.5 |  |
| x5c | Conditioned presence | Conditioned presence | Conditioned presence | Conditioned presence | 0 or 1 | Clause 5.1.8 | 3 |
| crit | Conditioned presence | Conditioned presence | Conditioned presence | Conditioned presence |  |  | 4 |
| sigT | shall be present | shall be present | shall be present | shall be present | 1 | Clause 5.2.1 | a |
| Service: signing a reference of the signing certificate | Conditioned presence | Conditioned presence | Conditioned presence | Conditioned presence | 1 |  | 3 |
|  SPO: x5t#256 | conditioned presence | conditioned presence | conditioned presence | conditioned presence | 0 or 1 | Clause 5.1.7 |  |
|  SPO: x5t#o | conditioned presence | conditioned presence | conditioned presence | conditioned presence | 0 or 1 | Clause 5.2.2 |  |
| sigD | may be present | may be present | may be present | may be present | 0 or 1 | Clause 5.2.8 |  |
| srAts | may be present | may be present | may be present | may be present | 0 or 1 | Clause 5.2.5 |  |
| srCm | may be present | may be present | may be present | may be present | ≥ 0 | Clause 5.2.3 |  |
| sigPl | may be present | may be present | may be present | may be present | 0 or 1 | Clause 5.2.4 |  |
| sigPId | may be present | may be present | may be present | may be present | 0 or 1 | Clause 5.2.7 |  |
| cSig | may be present | may be present | may be present | may be present | ≥ 0 | Clause 5.3.2 |  |
| adoTst | may be present | may be present | may be present | may be present | ≥ 0 | Clause 5.3.3 | 5 |
| sigPSt | may be present | may be present | may be present | may be present | 0 or 1 | Clause 5.3.7 | b |
| sigTst | \* | shall be present | shall be present | shall be present | B-B: ≥ 0 | Clause 5.3.4 | c, d5 |
| B-T, B-LT, B‑LTA: ≥ 1 |
| xVals | *\** | *\** | conditioned presence | conditioned presence | 0 or 1 | Clause 5.3.5.1 | e, 6 |
| xRefs | *\** | *\** | shall not be present | shall not be present | B-B, B-T: 0 or 1 | Clause A.1.1 | f, g |
| B-LT, B-LTA: 0 |
| axVals | \* | \* | conditioned presence | conditioned presence | 0 or 1 | Clause 5.3.5.3 | e, 7 |
| axRefs | *\** | *\** | shall not be present | shall not be present | B-B, B-T: 0 or 1 | Clause A.1.3 | f, g, h |
| B-LT, B-LTA: 0 |
| rVals | *\** | *\** | conditioned presence | conditioned presence | 0 or 1 | Clause 5.3.5.2 | i, j, 8 |
| rRefs | *\** | *\** | shall not be present | shall not be present | B-B, B-T: 0 or 1 | Clause A.1.2 |  |
| B-LT, B-LTA: 0 |
| arVals | *\** | *\** | conditioned presence | conditioned presence | 0 or 1 | Clause 5.3.5.4 | j, k, 9 |
| arRefs | *\** | *\** | shall not be present | shall not be present | B-B, B-T: 0 or 1 | Clause A.1.4 | h |
| B-LT, B-LTA: 0 |
| sigRTst | *\** | *\** | shall not be present | shall not be present | B-B, B-T: ≥ 0 | Clause A.1.5.1 |  |
| B-LT, B-LTA: 0 |
| rfsTst | *\** | *\** | shall not be present | shall not be present | B-B, B-T: ≥ 0 | Clause A.1.5.2 |  |
| B-LT, B-LTA: 0 |
| Service: Incorporation of validation data for electronic time-stamps | *\** | *\** | shall be provided | shall be provided | - | - | l, m10 |
|  SPO: tstVd | *\** | *\** | conditioned presence | conditioned presence | ≥ 0 | Clause 5.3.6.1 |  |
|  SPO: certificate and revocation values embedded in the electronic time-stamp itself | *\** | *\** | conditioned presence | conditioned presence | ≥ 0 | - |  |
| arcTst | \* | \* | \* | shall be present | ≥ 1 | Clause 5.3.6.2 | n, o |

Additional requirements:

1. Requirement for sigT. The generator shallinclude the claimed UTC time when the signature was generated as content of the sigT header parameter.
2. Requirement for sigPSt. This header parameter may be incorporated into the JAdES signature only if the sigPId is also incorporated and it contains the hashAV member with the digest value of the signature policy document. Otherwise the sigPSt shall not be incorporated into the JAdES signature.
3. Requirement for sigTst. Each sigTst header parameter shall contain only one electronic time-stamp.
4. Requirement for sigTst. The electronic time-stamps encapsulated within the sigTst header parameters shall be created before the signing certificate has been revoked or has expired
5. Requirement for xVals and axVals. Duplication of certificate values within the signature should be avoided.
6. Requirement for xRefs and axRefs. The references to certificates should not include the kid member.
7. Requirement for xRefs and axRefs. The references to certificates shall not include the x5u member.
8. Requirement for axRefs and arRefs. The axRefs and arRefs header parameters may be used when a at least an attribute certificate or a signed assertion is incorporated into the JAdES signature. Otherwise, axRefs and arRefs header parameters shall not be used.
9. Requirement for incorporation of rVals. If a JAdES-B-LT or a JAdES-B-LTA signature is generated, the incorporation of rVals shall be determined by the specification providing details on how to incorporate the header parameter to the underlying JSON Web Signature.
10. Requirement for rVals and arVals. Duplication of revocation values within the signature should be avoided.
11. Requirement for incorporation of arVals. If a JAdES-B-LT or a JAdES-B-LTA signature is generated, the incorporation of arVals shall be determined by the specification providing details on how to incorporate the header parameter to the underlying JSON Web Signature.
12. Requirement for service "incorporation of validation data for electronic time-stamps". The validation data for electronic time-stamps shall be present within the tstVd header parameter or embedded in the electronic time-stamp itself.
13. Requirement for service "incorporation of validation data for electronic time-stamps". The validation data for electronic time-stamps should be included in the tstVd header parameter.
14. Requirement for arcTst. Each arcTst header parameter may contain more than one electronic time-stamp issued by different TSAs.
15. Requirement for arcTst. Before generating and incorporating a new arcTst header parameter, all the validation material required for validating the JAdES signature shall be included. This validation material shall include all the certificates and all certificate status information (like CRLs or OCSP responses) required for:
* validating the signing certificate;
* validating the signing certificate of any cSignature incorporated into the signature;
* validating any attribute certificate or signed assertion present in the signature; and
* validating the signing certificate of any previous electronic time-stamp already incorporated into the signature within any JAdES electronic time-stamp container header parameter (including any arcTst).

NOTE 2: On cty, and ctys within sigD: see clauses 5.1.4 and 5.2.8.1 of the present document for details of their conditioned presence.

NOTE 3: On x5c and service "signing a reference of the signing certificate". Clauses 5.1.8, 5.1.7, and 5.2.2 specifies the conditions that decide the presence or absence of the x5c, x5t#S256, and x5t#o header parameters in a JAdES signature.

NOTE 4: On crit. Clause 5.1.9 specifies the conditions that decide the presence or absence of the crit header parameter in a JAdES signature.

NOTE 5: On sigTst, adoTst. Several instances of these header parameters can be incorporated into the JAdES signature, coming from different TSAs.

NOTE 6: On xVals. Clause 5.3.5.1 specifies the conditions that decide the presence or absence of the xVals element of etsiU JSON array in a JAdES signature.

NOTE 7: On axVals. Clause 5.3.5.3 specifies the conditions that decide the presence or absence of the axVals element of etsiU JSON array in a JAdES signature.

NOTE 8: On rVals. Clause 5.3.5.2 specifies the conditions that decide the presence or absence of the rVals element of etsiU JSON array in a JAdES signature.

NOTE 9: On arVals. Clause 5.3.5.4 specifies the conditions that decide the presence or absence of the arVals element of etsiU JSON array in a JAdES signature.

NOTE 10: On service "incorporation of validation data for electronic time-stamps": the incorporation of the validation material of the electronic time-stamps ensures that the JAdES signature actually contains all the validation material needed.

Annex A (normative):
Additional header parameters Specification

# A.1 Header parameters for validation data

## A.1.1 The xRefs header parameter

**Semantics**

The xRefs header parameter shall be an unsigned header parameter qualifying the signature.

The xRefs header parameter:

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.
3. May contain references to certificates in the path of the certificates used for signing the electronic time-stamps already incorporated into the signature when the xRefs unsigned header parameter is incorporated, including references to the electronic time-stamps' signing certificates and references to certificates of trust anchors if such certificates do exist.
4. May contain references to the certificates used to sign CRLs or OCSP responses for certificates referenced by references in 1) and 3), and references to certificates within their respective certificate paths. And
5. 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 srAts.

NOTE 1: The references to certificates exclusively used in the validation of attribute certificate or signed assertions are stored in the axRefs header parameter (see clause A.1.3).

**Syntax**

This header parameter shall be carried in the JWS Unprotected Header.

The xRefs member shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"x5Ids": {

 "type": "array",

 "items": {"$ref": "#/definitions/certId"},

 "minItems": 1

},

"certId":{

 "type": "object",

 "properties":{

 "digAlgVal": {"$ref": "#/definitions/digAlgVal"},

 "kid": {"type": "string", "contentEncoding" : "base64"},

 "x5u": {"type": "string", "format": "uri-reference"}

 },

 "required": ["digAlgVal"]

},

"xRefs": {"$ref": "#/definitions/x5Ids"},

The digAlgVal member has been already defined in clause 5.2.2 of the present document.

The content of kid member shall be the base64 encoding of one DER-encoded instance of type IssuerSerial type defined in IETF RFC 5035 [6].

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

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

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

If at least one of the following unsigned header parameters: xVals, axVals, or the arcTst, is incorporated into the signature, all the certificates referenced in xRefs shall be present elsewhere in the signature.

## A.1.2 The rRefs header parameter

**Semantics**

The rRefs header parameter shall be an unsigned header parameter that qualifies the signature.

The rRefs header parameter:

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 within the signing certificate path. 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.

1. May contain references to revocation values (e.g. CRLs or OCSP values) corresponding to certificates in the path of signing certificates of electronic time-stamps already incorporated into the signature when the rRefs unsigned header parameter is incorporated. It shall not contain references to revocation values for the trust anchors of these certificates.
2. May contain references to the revocation values corresponding to certificates used to sign CRLs or OCSP responses referenced in references from 1), 2) and 3), and to certificates within their respective certificate paths. And
3. 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 srAts header parameter.

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

References within rRefs header parameter may be references to CRLs, OCSP responses and other type of revocation data.

**Syntax**

This header parameter shall be carried in the JWS Unprotected Header.

The rRefs header parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"rRefs": {

 "type": "object",

 "properties": {

 "crlRefs": {

 "type": "array",

 "items": {

 "type": "object",

 "properties": {

 "digAlgVal": {"$ref": "#/definitions/digAlgVal"},

 "crlId": {

 "type": "object",

 "properties": {

 "issuer": {"type": "string", "contentEncoding" : "base64"},

 "issueTime": {"type": "string", "format": "date-time"},

 "number": {"type": "number"},

 "uri": {"type": "string", "format": "uri-reference"}

 },

 "required": ["issuer","issueTime"]

 }

 },

 "required": ["digAlgVal"]

 },

 "minItems": 1

 },

 "ocspRefs":{

 "type": "array",

 "items": {

 "type": "object",

 "properties": {

 "ocspId": {

 "type": "object",

 "properties": {

 "responderId": {

 "oneOf": [

 {"byName": {"type": "string", "contentEncoding" : "base64"}},

 {"byKey": {"type": "string", "contentEncoding" : "base64"}}

 ]

 },

 "producedAt": {"type": "string", "format": "date-time"},

 "uri": {"type": "string", "format": "uri-reference"}

 },

 "required": ["responderId", "producedAt"]

 },

 "digAlg": {"$ref": "#/definitions/digAlg"}

 },

 "required": ["ocspId","digAlg"]

 },

 "minItems": 1

 },

 "otherRefs": {

 "type": "array",

 "items": {"type":"object"},

 "minItems": 1

 }

 },

 "minProperties": 1

},

Empty rRefs header parameters shall not be incorporated.

The crlRefs member shall contain an array of references to CRLs.

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

The digAlgVal member of one item within the crlRefs array shall contain one indication of a digest algorithm, and the base64 encoding of the digest value of the DER-encoded referenced CRL.

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

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

The value of crlId‘s issuer member shall fulfil the requirements specified in RFC 1779 [12] for strings representing Distinguished Names.

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

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

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

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

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

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

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

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

The ocspId member of the items within the ocspRefs array shall include an identifier of the responder in its responderID member.

If the responder is identified by its name then this name shall appear within the responderID's byName member.

The value of byName member shall fulfil the requirements specified in RFC 1779 [12] for strings representing Distinguished Names.

If the responder is identified by the digest of the server's public key computed as mandated in IETF RFC 6960 [10], then the base64 encoding of the DER-encoded of byKey field specified in IETF RFC 6960 [10] shall appear within the responderID's byKey member.

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

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

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

NOTE 5: It is intended that this header parameter be used as a hint, as implementations can have alternative ways for retrieving the referenced OCSP response if it is not found at the referenced place.

The digAlgVal member of the items within the ocspRefs array shall contain one indication of a digest algorithm, and the base64 encoding of the DER-encoded OCSPResponse field defined in IETF RFC 6960 [10].

The digAlgVal member should be included within the OCSPRef member.

NOTE 6: The absence of the digAlgVal member of the items within the ocspRefs array makes OCSP responses substitutions attacks possible, if for instance OCSP responder keys are compromised. In this case, out-of-band mechanisms can be used to ensure that none of the OCSP responder keys have been compromised at the time of validation.

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

If at least one of the following unsigned header parameters: rVals, arVals, or the arcTst, is incorporated into the signature, all the revocation data referenced in rRefs shall be present elsewhere in the signature.

## A.1.3 The axRefs header parameter

**Semantics**

The axRefs header parameter shall be an unsigned header parameter that qualifies the signature.

The axRefs header parameter:

1. Shall contain, if they are not present within xRefs or x5t#o header parameters, the references to the trust anchors if certificates exist for them, and the references to CA certificates within the path of the signing certificate(s) of the attribute certificate(s) and signed assertion(s) incorporated into the JAdES signature. References present within xRefs or x5t#o header parameters should not be included.
2. Shall contain, if they are not present within xRefs or x5t#o header parameters, the reference(s) to the signing certificate(s) of the attribute certificate(s) and signed assertion(s) incorporated into the JAdES signature. References present within xRefs or x5t#o header parameters should not be included. And
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 into the JAdES signature. References present within xRefs or x5t#o header parameters should not be included.

**Syntax**

This header parameter shall be carried in the JWS Unprotected Header.

The axRefs header parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"attrCRefs": {"$ref": "#/definitions/x5Ids"}

If at least one of the following unsigned header parameters: xVals, axVals, or the arcTst, is incorporated into the signature, all the certificates referenced in axRefs shall be present elsewhere in the signature.

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

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

## A.1.4 The arRefs header parameter

**Semantics**

The arRefs header parameter shall be an unsigned header parameter that qualifies the signature.

The arRefs header parameter:

1. Shall contain, if they are not present within the rRefs header parameter, 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 into the JAdES signature. It shall not contain a revocation value for the trust anchors. References present within rRefs header parameter should not be included.

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

1. Shall contain, if they are not present within the rRefs header parameter, the references to the revocation value(s) for the signing certificate(s) of the attribute certificate(s) and signed assertion(s) incorporated into the JAdES signature. References present within rRefs header parameter should not be included. And
2. 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 into the JAdES signature. References present within rRefs header parameter should not be included.

**Syntax**

This header parameter shall be carried in the JWS Unprotected Header.

The arRefs header parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"arCRefs": {"$ref": "#/definitions/x5Ids"}

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

If at least one of the following unsigned header parameters: rVals, arVals, or the arcTst, is incorporated into the signature, all the revocation data referenced in arRefs shall be present elsewhere in the signature.

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

### A.1.5.1 The sigRTst header parameter

**Semantics**

The sigRTst header parameter shall be an unsigned header parameter qualifying the signature.

The sigRTst header parameter shall encapsulate electronic time-stamps on the JWS Signature Value, the signature time-stamp, if present, and the JAdES header parameters containing references to validation data.

**Syntax**

This header parameter shall be carried in the JWS Unprotected Header.

The sigRTst header parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"sigAndRfsTst": {"$ref": "#/definitions/tstContainer"}

This header parameter shall contain an electronic time-stamp that time-stamps the following JAdES components: the member encapsulating the JWS Signature Value, all present sigTst header parameters, xRefs, rRefs, and when present, axRefs, and arRefs.

#### A.1.5.1.2 Computation of the message imprint with Base64url incorporation

The message imprint computation input shall be the concatenation of the components, in the order they are listed below.

1. The value of the signature component, which is the base64url encoded JWS Signature Value.
2. The character '.'.
3. Those among the following unsigned header parameters that appear before sigRTst, in their order of appearance within the etsiU array, base64url-encoded, and separated by the character '.':
* The sigTst header parameters.
* The xRefs header parameter.
* The rRefs header parameter.
* The axRefs header parameter if it is present. And
* The arRefs header parameter if it is present.

#### A.1.5.1.3 Computation of the message imprint with JSON clear incorporation

The message imprint computation input shall be the concatenation of the components, in the order they are listed below.

1. The value of the signature component, which is the base64url encoded JWS Signature Value.
2. The character '.'.
3. Those among the following unsigned header parameters that appear before sigRTst, in their order of appearance within the etsiU array, canonicalized using the canonicalization algorithm identified in canonAlg member, and separated by the character '.'.
* The sigTst header parameters.
* The xRefs header parameter.
* The rRefs header parameter.
* The axRefs header parameter if it is present. And
* The arRefs header parameter if it is present.

### A.1.5.2 The rfsTst header parameter

#### A.1.5.2.1 Semantics and syntax

**Semantics**

The rfsTst header parameter shall be an unsigned header parameter qualifying the signature.

The rfsTst header parameter shall encapsulate electronic time-stamps on the JAdES header parameters containing references to validation data.

**Syntax**

This header parameter shall be carried in the JWS Unprotected Header.

The rfsTst header parameter shall be defined as in the JSON Schema file whose location is detailed in clause B.1, and is copied below for information.

"rfsTst": {"$ref": "#/definitions/tstContainer"}

This header parameter shall contain an electronic time-stamp that time-stamps the following JAdES header parameters: xRefs, rRefs, and when present, axRefs, and arRefs.

#### A.1.5.2.2 Computation of the message imprint with Base64url incorporation

The message imprint computation input shall be the concatenation of the components listed below, base64url encoded, and separated by the character '.', in their order of appearance within the etsiU array:

* The xRefs header parameter.
* The rRefs header parameter.
* The axRefs header parameter if it is present. And
* The arRefs header parameter if it is present.

#### A.1.5.2.3 Computation of the message imprint with clear JSON incorporation

The message imprint computation input shall be the concatenation of the components listed below, canonicalized using the canonicalization algorithm identified in canonAlg member, and separated by the character '.', in their order of appearance within the etsiU array.

* The xRefs header parameter.
* The rRefs header parameter.
* The axRefs header parameter if it is present. And
* The arRefs header parameter if it is present.

Annex B (normative):
JSON Schema file

# B.1 JSON Schema file location for JAdES header parameters

The file at <http://uri.etsi.org/19152/v1.1.1/JAdES19152v111-YYYYMM.xsd> (JAdES19182v111-YYYYMM.xsd) contains the definitions of the header parameters defined in the present document.

Annex C (informative):
Correspondence between XAdES tags and JAdES tags

## C.1 Correspondence between XAdES qualifying properties tags and JAdES header parameter tags

Table shows the correspondence between the tags used by the XAdES qualifying properties and the tags used by the JAdES header parameters.

Table 2: Correspondence between XAdES and JAdES tags

| XAdES tag | JAdES tag |
| --- | --- |
| SignedProperties | etsiSigProps |
| UnsignedProperties | etsiU |
| SigningTime | sigT |
| SigningCertificateV2 | x5t#o |
| X5Ids | x5Ids |
| SigaturePolicyIdentifier | sigPId |
| SignatureProductionPlaceV2 | sigPl |
| SignerRoleV2 | srAts |
| DataObjectFormat | sdF |
| AllDataObjectsTimeStamp | adoTst |
| CommitmentTypeIndication | srCm |
| CSignature | cSig |
| IndividualDataObjectsTimeStamp | idoTst |
| SignaturePolicyStore | sigPSt |
| SignatureTimeStamp | sigTst |
| OIdentifier | oId |
| EncapsualtedPKIDataType | pkiOb |
| ArchiveTimeStamp | arcTst |
| RefsOnlyTimeStampV2 | rfsTst |
| SigAndRefsTimeStampV2 | sigRTst |
| CertificateValues | xVals |
| RevocationValues | rVals |
| AttrAuthoritiesCVals | axVals |
| AttributeRevocationValues | arVals |
| TimeStampValidationData | tstVd |
| CompleteCertificateRefs | xRefs |
| RevocationRefs | rRefs |
| AttributeCertificateRefsV2 | axRefs |
| AttributeRevocationRefs | arRefs |

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

There may be mechanisms to achieve long-term availability and integrity of validation data different from the ones described in clause 5.3.6.

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

1. The clear specification of the semantics and syntax of the header parameter including its unique identifier.
2. The strategy of how this mechanism guarantees that all necessary parts of the signature are protected by this header parameter.
3. The strategy of how to handle signatures containing header parameters defined in the present document.

EXAMPLE: The objects defined in IETF RFC 4998 [i.11], annex A are examples of such alternative mechanisms but they only handle points 1) and 2).

Annex E (informative):
Bibliography

[1] Juan Carlos Cruellas:"Bringing JSON signatures to ETSI AdES framework: meet JAdES signatures". Paper submitted to Computer Standards & Interfaces.

NOTE: The contents of this document, as well as the draft proposals for ETSI TS 119 152-1-2 and ETSI TS 119 152-2, are based on the contents of the aforementioned paper. At the moment of producing this draft proposal, the resolution by Computer Standards & Interfaces on whether to publish it or not has not been yet taken.

Annex F (informative):
Change History

| Date | Version | Information about changes |
| --- | --- | --- |
| 2020 January | 0.0.1 | Version based on previous version circulated in October 2019 at ESI 68 (with a wrong TS number) after amendments of relevant parts. |
|  |  |  |
|  |  |  |
|  |  |  |

# History

|  |
| --- |
| **Document history** |
| <Version> | <Date> | <Milestone> |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

*Latest changes made on 2018-02-09*