Profile JSON Web Signature for OASIS Digital Signature Services Version 2.0

Working Draft 01

DD Month YYYY

Technical Committee:

[OASIS Digital Signature Services eXtended (DSS-X) TC](https://www.oasis-open.org/committees/dss-x/)

Chairs:

Stefan Hagen ([stefan@hagen.link](mailto:stefan@hagen.link)), Individual

Editor:

Andreas Kuehne ([kuehne@trustable.de](mailto:kuehne@trustable.de)), Individual

Stefan Hagen ([stefan@hagen.link](mailto:stefan@hagen.link)), Individual

Additional artifacts:

This prose specification is one component of a Work Product that also includes:

* JSON and XML schemas: <http://docs.oasis-open.org/dss-x/dss-core/v2.0/csd01/schemas/>

Related work:

This specification is related to:

* RFC 7515 JSON Web Signature (JWS): <https://tools.ietf.org/html/rfc7515>

Declared XML namespaces:

* http://docs.oasis-open.org/dss/ns/jws

Abstract:

This document defines a protocol and processing profile of the DSS Signing and Verifying Protocol specified in **[DSSCore]**, which allows to create JSON Web Signatures and to return verification outcome for each signature in a verification request.

Status:

This [Working Draft](https://www.oasis-open.org/policies-guidelines/tc-process) (WD) has been produced by one or more TC Members; it has not yet been voted on by the TC or [approved](https://www.oasis-open.org/policies-guidelines/tc-process) as a Committee Draft (Committee Specification Draft or a Committee Note Draft). The OASIS document [Approval Process](https://www.oasis-open.org/policies-guidelines/tc-process) begins officially with a TC vote to approve a WD as a Committee Draft. A TC may approve a Working Draft, revise it, and re-approve it any number of times as a Committee Draft.

Any machine-readable content ([Computer Language Definitions](https://www.oasis-open.org/policies-guidelines/tc-process)) declared Normative for this Work Product must also be provided in separate plain text files. In the event of a discrepancy between such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

URI patterns:

Initial publication URI:  
<http://docs.oasis-open.org/dss-x/>???

Permanent “Latest version” URI:  
http://docs.oasis-open.org/dss-x/???

(Managed by OASIS TC Administration; please don’t modify.)

Copyright © OASIS Open 2017. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full [Policy](https://www.oasis-open.org/policies-guidelines/ipr) may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Table of Contents

[1 Introduction 4](#_Toc511147236)

[1.1 Organization of DSS Core Protocols, Elements, and Bindings 4](#_Toc511147237)

[1.2 Terminology 4](#_Toc511147238)

[1.2.1 Terms and Definitions 4](#_Toc511147239)

[1.2.2 Abbreviated Terms 4](#_Toc511147240)

[1.3 Normative References 4](#_Toc511147241)

[1.4 Non-Normative References 5](#_Toc511147242)

[1.5 Typographical Conventions 5](#_Toc511147243)

[2 Design Considerations 7](#_Toc511147244)

[2.1 Construction Principles 7](#_Toc511147245)

[2.2 Domain Models 7](#_Toc511147246)

[2.2.1 Date and Time Model 7](#_Toc511147247)

[2.3 Schema Organization and Namespaces 7](#_Toc511147248)

[2.4 DSS Overview (Non-normative) 8](#_Toc511147249)

[2.5 Syntax variants 8](#_Toc511147250)

[3 Structure Models 9](#_Toc511147251)

[3.1 Structure Models defined in this document 9](#_Toc511147252)

[3.1.1 Component SerializationSyntax 9](#_Toc511147253)

[3.1.1.1 Semantics 9](#_Toc511147254)

[3.1.1.2 XML Syntax 9](#_Toc511147255)

[3.1.1.3 JSON Syntax 9](#_Toc511147256)

[3.1.2 Component OptionalInputSign 10](#_Toc511147257)

[3.1.2.1 Semantics 10](#_Toc511147258)

[3.1.2.2 XML Syntax 11](#_Toc511147259)

[3.1.2.3 JSON Syntax 12](#_Toc511147260)

[3.1.3 Component OptionalInputVerify 13](#_Toc511147261)

[3.1.3.1 Semantics 13](#_Toc511147262)

[3.1.3.2 XML Syntax 13](#_Toc511147263)

[3.1.3.3 JSON Syntax 13](#_Toc511147264)

[4 Conformance 15](#_Toc511147265)

[4.1 Level 1 – “Basic” 15](#_Toc511147266)

[4.2 Level 2 – “Comprehensive” 15](#_Toc511147267)

[Appendix A. Index 16](#_Toc511147268)

[Appendix B. Revision History 17](#_Toc511147269)

# Introduction

## Organization of DSS Core Protocols, Elements, and Bindings

This document defines a protocol and processing profile of the DSS Signing and Verifying Protocol specified in **[DSSCore]**, which allows to support the creation and verification of JSON Web Signatures.

The following sections describe how to understand the rest of this document.

## Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

### Terms and Definitions

For the purposes of this document, the following applies:

**Term** — meaning and maybe ref

### Abbreviated Terms

**Acronym** — Spelled out

## Normative References

[RFC2119] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.

**[RFC 2396]** T. Berners-Lee et al. *Uniform Resource Identifiers (URI): Generic Syntax.* IETF RFC 2396, August 1998.   
<http://www.ietf.org/rfc/rfc2396.txt>.

[DSS2XSD] S. Hagen,. *DSS 2.0 Core Schema*. OASIS, ToDo.

[DSS2BASEXSD] S. Hagen,. *DSS 2.0 Base Schema*. OASIS, ToDo.

**[DSSJWS-XSD]** A. Kuehne, et. al.: “*DSS JSON Web Signatures Schema”*, , ToDo.

**[RFC 2440]** J. Callas, L. Donnerhacke, H. Finney, R. Thayer. *OpenPGP Message Format*. IETF RFC 2440, November 1998.  
<http://www.ietf.org/rfc/rfc2440.txt>.

**[RFC 2616]** R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. IETF RFC 2616, June 1999.  
[http://www.ietf.org/rfc/rfc2616.txt](http://www.ietf.org/rfc/rfc2440.txt).

**[RFC 2648]** R. Moats. *A URN Namespace for IETF Documents*. IETF RFC 2648, August 1999.   
<http://www.ietf.org/rfc/rfc2648.txt>.

**[RFC 2822]** P. Resnick. *Internet Message Format*. IETF RFC 2822, April 2001. <http://www.ietf.org/rfc/rfc2822.txt>

**[RFC 3161]** C. Adams, P. Cain, D. Pinkas, R. Zuccherato. *Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)*. IETF RFC 3161, August 2001.   
[http://www.ietf.org/rfc/rfc3161.txt](http://www.ietf.org/rfc/rfc3075.txt).

**[RFC 5280]** D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. IETF RFC 5280, May 2008.   
<http://www.ietf.org/rfc/rfc5280.txt>.

**[RFC 5652]** R. Housley. *Cryptographic Message Syntax*. IETF RFC 5652, September 2009.  
<http://www.ietf.org/rfc/rfc5652.txt>.   
(Remark: As used in DSS, all implementations based upon RFC 5652 and previous releases of CMS will suffice. For the sake of simplicity the "urn:ietf:rfc:3369" is used throughout the document to indicate a CMS message as specified in RFC 5652 or RFC 3369 or any version (including PKCS #7).

**[****RFC7159]** T. Bray, Ed., Google, Inc., The JavaScript Object Notation (JSON) Data Interchange Format, ISSN: 2070-1721, March 2014.   
<https://tools.ietf.org/html/rfc7159>.

**[RFC7515]** M. Jones, Microsoft, JSON Web Signature (JWS), ISSN: 2070-1721, May 2015.   
<https://tools.ietf.org/html/rfc7515>.

[XML] Extensible Markup Language (XML) 1.0 (Fifth Edition), T. Bray, J. Paoli, M. Sperberg-McQueen, E. Maler, F. Yergeau, Editors, W3C Recommendation, November 26, 2008, <http://www.w3.org/TR/2008/REC-xml-20081126/>.   
Latest version available at <http://www.w3.org/TR/xml>.

[XML-Schema-1] W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures, S. Gao, M. Sperberg-McQueen, H. Thompson, N. Mendelsohn, D. Beech, M. Maloney, Editors, W3C Recommendation, April 5, 2012,   
<http://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/>.   
Latest version available at <http://www.w3.org/TR/xmlschema11-1/>.

[XML-Schema-2] W3C XML Schema Definition Language (XSD) 1.1 Part 2: DatatypesW3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes, D. Peterson, S. Gao, A. Malhotra, M. Sperberg-McQueen, H. Thompson, Paul V. Biron, Editors, W3C Recommendation, April 5, 2012,   
<http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/>.   
Latest version available at <http://www.w3.org/TR/xmlschema11-2/>.

**[XPATH]** XML Path Language (XPath) Version 1.0. W3C Recommendation 16 November 1999 <http://www.w3.org/TR/xpath>

## Non-Normative References

[ISO8601] Data elements and interchange formats — Information interchange — Representation of dates and times, International Standard, ISO 8601:2004(E), December 1, 2004, <https://www.iso.org/standard/40874.html>.

## Typographical Conventions

Keywords defined by this specification use this monospaced font.

Normative source code uses this paragraph style.

Text following the special symbol («) – an opening Guillemet (or French quotation mark) – within this specification identifies conformance statements. Every conformance statement is separated from the following text with the special end symbol (») – a closing Guillemet, and has been assigned a reference that follows that end symbol in the format [dSS-section#-local#].

Some sections of this specification are illustrated with non-normative examples.

Example 1: text describing an example uses this paragraph style

Non-normative examples use this paragraph style.

All examples in this document are non-normative and informative only.

Representation-specific text is indented and marked with vertical lines.

Representation-Specific Headline

Normative representation-specific text

All other text is normative unless otherwise labeled e.g. like:

Non-normative Comment:

This is a pure informative comment that may be present, because the information conveyed is deemed useful advice or common pitfalls learned from implementer or operator experience and often given including the rationale.

# JSON Web Signature profile

## Overview

This profile supports operations within each phase of the lifecycle of JWS styled digital signatures encoded in the JSON format defined in [RFC7515].

For the generation of JWS signatures, the following operations apply:

* SignRequest: This structure supports requests for:
  + Requesting JWS signature as defined in [RFC7515].
* SignResponse: This structure supports delivery of:
  + Created (co-)signature as defined in [RFC7515].

For advanced signature verification (and updating) the following operations apply:

* VerifyRequest: This structure supports requests for:
  + Verifying an existing JWS signature forms as defined in [RFC7515].
* VerifyResponse: This structure supports delivery of:
  + Signature verification result of a JWS signature.
  + An updated signature if requested.

## Profile Features

### Scope

This document profiles the DSS signing and verifying protocols defined in [DSSCore]. Detached content (as described in Appendix F. of **[RFC 7515])** is not supported by this profile.

### Relationship to Other Profiles

The profile in this document is based on the [DSSCore].

### Signature Object

This profile supports the creation and verification of signatures as defined in [RFC7515] in the JWS Compact Serialization (defined in [RFC7515], section 7.1) and JWS JSON Serialization format (defined in [RFC7515], section 7.2).

### Request Syntax

The signature format may indicate the transport syntax to be JSON. But this profile does not limit the applied transport syntax in any way.

## Profile of Signing Protocol

The present profile allows requesting signatures as defined in [RFC7515].

### Component SignRequest

This clause profiles the SignRequest component.

#### Component InputDocuments

The JWS signature includes the payload within the resulting artifact. So, the server MUST be able to access the document directly. Hashed or transformed documents MUST NOT be provided.

##### Component Document

This component MUST be present.

This component contains the content to be signed.

In case of adding an additional JWS signature then this component contains the existing signature. The payload to be signed is included within the existing signature.

##### Component DocumentHash

This component MUST NOT be present.

##### Component TransformedData

This component MUST NOT be present.

#### Component OptionalInputs

##### New Optional Inputs

###### Optional Input OptionalInputSign

To handle the specifics of a request for a JWS signature this profile introduces an additional element within the OptionalInputsSign component. This component includes the optional element JWSProfileInputs that MUST satisfy the requirements specified in section ‘Component OptionalInputSign’.

XML Syntax

The XML type OptionalInputsSignType is extended with the element JWSProfileInput.

The OptionalInputsSignType XML element is defined in XML Schema file [DSSCORE\_SCHEMA]. The extension of that element is copied below for information.

<xs:complexType name="OptionalInputsSignType">

[…]

<xs:choice>

[…}

<xs:element name="JWSProfileInputs" type="jws:OptionalInputSignType"/>

[…}

</xs:choice>

[…}

</xs:complexType>

The XML syntax of the JWSProfileInputs element is specified within section ‘Component OptionalInputVerify’.

JSON Syntax

The OptionalInputsSignType JSON object is extended with the element JWSProfileInput.

The SerializationSyntaxType JSON object SHALL be defined as in JSON Schema file [JSON SCHEMA FILE NAME] whose location is detailed in clause [CLAUSE FOR LINK TO THE JSON SCHEMA FILE], and is copied below for information.

"dss2-OptionalInputsSignType": {

"type": "object",

"properties": {

[…}

"optJws": {

"ref": "#/definitions/jws-OptionalInputSignType"

}

}

}

The name of the single property introduced in the JSON schema snippet above maps to the element’s name shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| JWSProfileInputs | optJws | [] |

##### Optional Inputs already defined in the Core

Most of the optional inputs specified in the [DSS Core] are usable in this profile. See restrictions and constrains on the defined optional inputs below.

###### Optional Input SignatureType

This component is OPTIONAL. If present, SignatureType SHALL be:

**urn:ietf:rfc:7515:signature**

for requesting JWS digital signatures, or

**urn:ietf:rfc:7515:mac**

for requesting JWS MAC computation.

###### Optional input KeySelector

The requester MAY request to the server to select specific key material. This can be done by using the KeySelector optional input. In this case, the element KeyName of the KeySelector Optional Input MAY contain the key ID (‘kid’).

###### Optional Input SignedProperties

The requester MAY request the addition of optional ‘protected headers’ using the SignedProperties element’s Property child. These additional headers MUST NOT have the same names as headers used by the server (e.g. the ‘alg’ header). See section 4.2. of [RFC7515] for details.

If at least one header name is included more than once in the set or collides with a name already used as header an error will be returned. The ResultMajor value of

urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError

and a ResultMinor value of

urn:oasis:names:tc:dss-x:2.0:profiles:jws:HeaderNameCollision

will mark this condition.

###### Optional Input UnsignedProperties

When using the variants of the JWS JSON Serialization syntax the requester MAY request the addition of optional ‘unprotected headers’ using the UnsignedProperties element’s Property child. These additional headers MUST NOT have the same names as headers used by the server (e.g. the ‘alg’ header). See section 4.2. of [RFC7515] for details.

In case the serialization syntax does not support unprotected headers a ResultMajor value of

urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError

and a ResultMinor value of

urn:oasis:names:tc:dss-x:2.0:profiles:jws:UnprotectedHeaderNotSupported

will be returned in the Result element of the response.

If at least one header name is included more than once in the set or collides with a name already used as header an error will be returned. The ResultMajor value of

urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError

and a ResultMinor value of

urn:oasis:names:tc:dss-x:2.0:profiles:jws:UnprotectedHeaderNameCollision

will mark this condition.

###### Optional Input AddTimestamp

The JWS specification does not define specific mechanism for long term validation. Nevertheless, it is possible to add timestamps as unprotected header. Therefore, the optional inputs AddTimestamp MAY be used.

###### Optional Input SignedReferences, SignaturePlacement and IncludeObject

These optional inputs MUST NOT be used in the context of this profile.

### Component SignResponse

This clause profiles the SignResponse component.

#### Component SignatureObject

This component contains the created or updated signature in the Base64Signature element.

If JWS Compact Serialization is used (Section 3.1 RFC7515), it represents the last part of the concatenation, being:

* BASE64URL(JWS Signature)

If JWS JSON Serialization is used (Section 3.2 RFC7515), it represents

* the “signature” member, with the value BASE64URL(JWS Signature)

#### Optional Outputs

None of the optional outputs specified in the [DSS Core] are neither precluded nor further profiled in this profile.

## Profile of Verifying Protocol

### Component VerifyRequest

This clause specifies the profile for the contents of the VerifyRequest when used for requesting verification of JWS signatures.

#### Component SignatureObject

This component MUST contain the JWS in the Base64Signature element. The type of signature, specified by the MimeType element of the Base64DataType component, MUST be “application/jose” or “application/jose+json”.

#### Component InputDocument

As the JWS signature encapsulates the payload the InputDocument element MUST NOT be used.

#### Component OptionalInputs

Most of the optional inputs specified in the **[DSS Core]** are usable in this profile. See restrictions and constrains on the defined optional inputs below.

##### New Optional Inputs

###### Optional Input OptionalInputVerify

To handle the specifics of a verification request for a JWS signature this profile introduces an additional element within the OptionalInputsVerify component. This component includes the optional element JWSProfileInputs that MUST satisfy the requirements specified in section ‘Component OptionalInputVerify’.

XML Syntax

The XML type OptionalInputsSignType will be extended with the element JWSProfileInput.

The OptionalInputsSignType XML element is defined in XML Schema file [DSSCORE\_SCHEMA]. The extension of that element is copied below for information.

<xs:complexType name="OptionalInputsVerifyType">

[…]

<xs:choice>

[…}

<xs:element name="JWSProfileInputs" type="jws:OptionalInputVerifyType"/>

[…}

</xs:choice>

[…}

</xs:complexType>

The XML syntax of the JWSProfileInputs element is specified within section ‘Component OptionalInputVerify’.

JSON Syntax

The OptionalInputsSignType JSON object will be extended with the element JWSProfileInput.

The SerializationSyntaxType JSON object SHALL be defined as in JSON Schema file [JSON SCHEMA FILE NAME] whose location is detailed in clause [CLAUSE FOR LINK TO THE JSON SCHEMA FILE], and is copied below for information.

"dss2-OptionalInputsVerifyType": {

"type": "object",

"properties": {

[…}

"optJws": {

"ref": "#/definitions/jws-OptionalInputVerifyType"

}

}

}

The name of the single property introduced in the JSON schema snippet above maps to the element’s name shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| JWSProfileInputs | optJws | [] |

##### Optional Inputs already defined in the Core

Most of the optional inputs specified in the [DSS Core] are usable in this profile. See restrictions and constrains on the defined optional inputs in this section.

###### Optional Input ReturnUpdatedSignature and ReturnTimestampedSignature

The JWS specification does not define specific mechanism for long term validation. Nevertheless, it is possible to add CRLs, OCSP responses and timestamps as unprotected header. Therefore, the mentioned optional inputs ReturnUpdatedSignature and ReturnTimestampedSignature MAY be used.

###### Optional Input ReturnTransformedDocument

The JWS specification does not define any transformation. This optional input element MUST NOT be used.

### Component VerifyResponse

This clause profiles the dss:VerifyResponse component.

#### Component OptionalOutputs

None of the optional outputs specified in the [DSS Core] are precluded in this profile. This section constrains some of them.

##### New Optional Inputs

###### Optional Input OptionalOutputsVerify

To handle the specifics of a verification request for a JWS signature this profile introduces an additional element within the OptionalOutputsVerify component. This component includes the optional element JWSProfileOutputs that MUST satisfy the requirements specified in section ‘Component OptionalOutputVerify’.

XML Syntax

The XML type OptionalOutputsSignType will be extended with the element JWSProfileInput.

The OptionalOutputsSignType XML element is defined in XML Schema file [DSSCORE\_SCHEMA]. The extension of that element is copied below for information.

<xs:complexType name="OptionalOutputsVerifyType">

[…]

<xs:choice>

[…}

<xs:element name="JWSProfileOutputs" type="jws:OptionalOutputVerifyType"/>

[…}

</xs:choice>

[…}

</xs:complexType>

The XML syntax of the JWSProfileOutputs element is specified within section ‘Component OptionalOutputVerify’.

JSON Syntax

The OptionalOutputsVerifyType JSON object will be extended with the element JWSProfileOutput.

The SerializationSyntaxType JSON object SHALL be defined as in JSON Schema file [JSON SCHEMA FILE NAME] whose location is detailed in clause [CLAUSE FOR LINK TO THE JSON SCHEMA FILE], and is copied below for information.

"dss2-OptionalOutputsVerifyType": {

"type": "object",

"properties": {

[…}

"optJws": {

"ref": "#/definitions/jws-OptionalOutputVerifyType"

}

}

}

The name of the single property introduced in the JSON schema snippet above maps to the element’s name shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| JWSProfileOutputs | optJws | [] |

##### Optional Inputs already defined in the Core

Most of the optional inputs specified in the [DSS Core] are usable in this profile. See restrictions and constrains on the defined optional inputs below.

###### Optional Output UpdatedSignature

This component MAY contain a SignatureObject element.

# Structure Models

## Structure Models defined in this document

The XML elements of this section are defined in the XML namespace 'http://docs.oasis-open.org/dss/ns/jws'.

[namespace http://docs.oasis-open.org/dss/ns/jws explanation]

### Component SerializationSyntax

#### Semantics

The SerializationSyntax component requests the syntax of the JWS signature as defined in section 7.1, 7.2.1 and 7.2.2 in [RFC7515].

Below follows a list of the sub-components that MAY be present within this component:

* The value element MUST contain one instance of a URI. Its value is limited to an item of the following set:  
  urn:ietf:rfc:7515:compact  
  urn:ietf:rfc:7515:general  
  urn:ietf:rfc:7515:flattened  
  [sub component value details]

Non-normative Comment:

[component SerializationSyntax non normative details]

#### XML Syntax

The XML type SerializationSyntaxType SHALL implement the requirements defined in the SerializationSyntax component.

The SerializationSyntaxType XML element SHALL be defined as in XML Schema file [FILE NAME] whose location is detailed in clause [CLAUSE FOR LINK TO THE XSD], and is copied below for information.

<xs:simpleType name="SerializationSyntaxType">

<xs:restriction base="xs:anyURI">

<xs:enumeration value="urn:ietf:rfc:7515:compact"/>

<xs:enumeration value="urn:ietf:rfc:7515:general"/>

<xs:enumeration value="urn:ietf:rfc:7515:flattened"/>

</xs:restriction>

</xs:simpleType>

Each child element of SerializationSyntaxType XML element SHALL implement in XML syntax the sub-component that has a name equal to its local name. [component SerializationSyntax XML schema details]

#### JSON Syntax

The SerializationSyntaxType JSON object SHALL implement in JSON syntax the requirements defined in the SerializationSyntax component.

The SerializationSyntaxType JSON object SHALL be defined as in JSON Schema file [JSON SCHEMA FILE NAME] whose location is detailed in clause [CLAUSE FOR LINK TO THE JSON SCHEMA FILE], and is copied below for information.

"jws-SerializationSyntaxType": {

"$xsd-full-type": "jws:SerializationSyntaxType",

"type": "string",

"enum": ["urn:ietf:rfc:7515:compact", "urn:ietf:rfc:7515:general", "urn:ietf:rfc:7515:flattened"]

}

Properties in the JSON schema above SHALL implement sub-component of SerializationSyntax component mapped by names as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| value | value | [] |

[component SerializationSyntax JSON schema details]

### Component OptionalInputSign

#### Semantics

This component is added to the DSS-X Core’s OptionalInputSign component and contains all elements introduced by this profile related to signature requests.

Below follows a list of the sub-components that MAY be present within this component:

* The optional SerializationSyntax element MUST contain a sub-component. A given element MUST satisfy the requirements specified in section SerializationSyntax. [sub component SerializationSyntax details]
* The optional JWSMediaType element MUST contain a string. The mime type of the signature MAY be indicated using this element. The media type will be reflected by the ‘typ’ header of the signature as defined in section 4.1.9 in [RFC7515].
* The optional JWSContentType element MUST contain a string. The mime type of the secured content (the payload) MAY be indicated using this element. If omitted, the MimeType value of the input document will be used as content type. If provided the content media type will be reflected in the ‘cty’ header of the signature as defined in 4.1.9 in [RFC7515].
* The optional CriticalHeader element MAY occur zero or more times containing a string. This element MAY define protected header names that will be added into the ‘crit’ header. See section 4.1.11. in [RFC7515]. If the at least one header name does not exist in the set of protected headers an error will be returned. The a ResultMajor value of urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError and a ResultMinor value of urn:oasis:names:tc:dss-x:2.0:profiles:jws:CriticalHeaderNotFound will mark this condition.
* The optional AppendSignature element MUST contain a boolean. Its default value is 'false'. The value ‘true’ will instruct the server to append an additional signature element to an existing ‘signatures’ array according to the ‘General JWS JSON Serialization’ syntax as defined in section 7.2.1. in [RFC7515]. In this case, the pre-existing JWS signature MUST be transported in the InputDocuments / Document component and the SerializationSyntax element MUST have the value of urn:ietf:rfc:7515:general . If the input signature is not of the appropriate syntax or the SerializationSyntax element contains another value than expected an error will be returned. The ResultMajor value of urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError and a ResultMinor value of urn:oasis:names:tc:dss-x:2.0:profiles:jws:InappropriateSerializationSyntax will mark this condition.

Non-normative Comment:

[component OptionalInputSign non normative details]

#### XML Syntax

The XML type OptionalInputSignType SHALL implement the requirements defined in the OptionalInputSign component.

The OptionalInputSignType XML element SHALL be defined as in XML Schema file [FILE NAME] whose location is detailed in clause [CLAUSE FOR LINK TO THE XSD], and is copied below for information.

<xs:complexType name="OptionalInputSignType">

<xs:sequence>

<xs:choice>

<xs:element maxOccurs="1" minOccurs="0" name="SerializationSyntax" type="jws:SerializationSyntaxType"/>

<xs:element maxOccurs="1" minOccurs="0" name="JWSMediaType" type="xs:string"/>

<xs:element maxOccurs="1" minOccurs="0" name="JWSContentType" type="xs:string"/>

<xs:element maxOccurs="unbounded" minOccurs="0" name="CriticalHeader" type="xs:string"/>

<xs:element default="false" maxOccurs="1" minOccurs="0" name="AppendSignature" type="xs:boolean"/>

</xs:choice>

</xs:sequence>

</xs:complexType>

Each child element of OptionalInputSignType XML element SHALL implement in XML syntax the sub-component that has a name equal to its local name. [component OptionalInputSign XML schema details]

#### JSON Syntax

The OptionalInputSignType JSON object SHALL implement in JSON syntax the requirements defined in the OptionalInputSign component.

The OptionalInputSignType JSON object SHALL be defined as in JSON Schema file [JSON SCHEMA FILE NAME] whose location is detailed in clause [CLAUSE FOR LINK TO THE JSON SCHEMA FILE], and is copied below for information.

"jws-OptionalInputSignType": {

"$xsd-full-type": "jws:OptionalInputSignType",

"type": "object",

"properties": {

"jwscontentType": {

"type": "string"

},

"jwsmediaType": {

"type": "string"

},

"serialType": {

"type": "string"

},

"jwsMediaType": {

"type": "string"

},

"jwsContentType": {

"type": "string"

},

"critHeader": {

"type": "array",

"items": {

"type": "string"

}

},

"appSig": {

"type": "boolean",

"default": "false"

}

}

}

Properties in the JSON schema above SHALL implement sub-component of OptionalInputSign component mapped by names as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| SerializationSyntax | serialType | [] |
| JWSMediaType | jwsMediaType | [] |
| JWSContentType | jwsContentType | [] |
| CriticalHeader | critHeader | [] |
| AppendSignature | appSig | [] |

[component OptionalInputSign JSON schema details]

### Component OptionalInputVerify

#### Semantics

This component is added to the DSS-X Core’s OptionalInputVerify component and contains all elements introduced by this profile related to verification requests.

Below follows a list of the sub-components that MAY be present within this component:

* The optional JWSSignatureIndex element MAY occur zero or more times containing a non-negative integer. If present, the integer value is a zero-based index of the ‘signatures’ array according to the ‘General JWS JSON Serialization’ syntax as defined in section 7.2.1. in [RFC7515]. This element addresses the signatures to be verified. If the signature index is less than zero or bigger or equal to the length of the ‘signatures’ array, then an error will be returned. The a ResultMajor value of urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError and a ResultMinor value of urn:oasis:names:tc:dss:1.0:resultminor:InvalidIndex will mark this condition.

Non-normative Comment:

[component OptionalInputVerify non normative details]

#### XML Syntax

The XML type OptionalInputVerifyType SHALL implement the requirements defined in the OptionalInputVerify component.

The OptionalInputVerifyType XML element SHALL be defined as in XML Schema file [FILE NAME] whose location is detailed in clause [CLAUSE FOR LINK TO THE XSD], and is copied below for information.

<xs:complexType name="OptionalInputVerifyType">

<xs:sequence>

<xs:choice>

<xs:element maxOccurs="unbounded" minOccurs="0" name="JWSSignatureIndex" type="xs:nonNegativeInteger"/>

</xs:choice>

</xs:sequence>

</xs:complexType>

Each child element of OptionalInputVerifyType XML element SHALL implement in XML syntax the sub-component that has a name equal to its local name. [component OptionalInputVerify XML schema details]

#### JSON Syntax

The OptionalInputVerifyType JSON object SHALL implement in JSON syntax the requirements defined in the OptionalInputVerify component.

The OptionalInputVerifyType JSON object SHALL be defined as in JSON Schema file [JSON SCHEMA FILE NAME] whose location is detailed in clause [CLAUSE FOR LINK TO THE JSON SCHEMA FILE], and is copied below for information.

"jws-OptionalInputVerifyType": {

"$xsd-full-type": "jws:OptionalInputVerifyType",

"type": "object",

"properties": {

"jwssignatureIndex": {

"type": "array",

"items": {

"type": "integer"

}

},

"jwsSigIdx": {

"type": "array",

"items": {

"type": "integer"

}

}

}

}

Properties in the JSON schema above SHALL implement sub-component of OptionalInputVerify component mapped by names as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| JWSSignatureIndex | jwsSigIdx | [] |

[component OptionalInputVerify JSON schema details]

### Component OptionalOutputVerify

#### Semantics

This component is added to the DSS-X Core’s OptionalOutputVerify component and contains all elements introduced by this profile related to verification responses.

Below follows a list of the sub-components that MAY be present within this component:

* The optional JWSVerifiedSignatureIndex element MAY occur zero or more times containing a non-negative integer. This set of integer value is a zero-based index of the ‘signatures’ array according to the ‘General JWS JSON Serialization’ syntax as defined in section 7.2.1. in [RFC7515]. Each item marks the successful verification of the addressed signature.

Non-normative Comment:

[component OptionalOutputVerify non normative details]

#### XML Syntax

The XML type OptionalOutputVerifyType SHALL implement the requirements defined in the OptionalOutputVerify component.

The OptionalOutputVerifyType XML element SHALL be defined as in XML Schema file [FILE NAME] whose location is detailed in clause [CLAUSE FOR LINK TO THE XSD], and is copied below for information.

<xs:complexType name="OptionalOutputVerifyType">

<xs:sequence>

<xs:choice>

<xs:element maxOccurs="unbounded" minOccurs="0" name="JWSVerifiedSignatureIndex" type="xs:nonNegativeInteger"/>

</xs:choice>

</xs:sequence>

</xs:complexType>

Each child element of OptionalOutputVerifyType XML element SHALL implement in XML syntax the sub-component that has a name equal to its local name. [component OptionalOutputVerify XML schema details]

#### JSON Syntax

The OptionalOutputVerifyType JSON object SHALL implement in JSON syntax the requirements defined in the OptionalOutputVerify component.

The OptionalOutputVerifyType JSON object SHALL be defined as in JSON Schema file [JSON SCHEMA FILE NAME] whose location is detailed in clause [CLAUSE FOR LINK TO THE JSON SCHEMA FILE], and is copied below for information.

"jws-OptionalOutputVerifyType": {

"$xsd-full-type": "jws:OptionalOutputVerifyType",

"type": "object",

"properties": {

"jwsverifiedSignatureIndex": {

"type": "array",

"items": {

"type": "integer"

}

},

"jwsVerifiedIdx": {

"type": "array",

"items": {

"type": "integer"

}

}

}

}

Properties in the JSON schema above SHALL implement sub-component of OptionalOutputVerify component mapped by names as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| JWSVerifiedSignatureIndex | jwsVerifiedIdx | [] |

[component OptionalOutputVerify JSON schema details]

# Identifiers defined in this specification

## Result Identifiers

This profile defines the <ResultMinor> values listed below:

urn:oasis:names:tc:dss-x:2.0:profiles:jws:CriticalHeaderNotFound

urn:oasis:names:tc:dss-x:2.0:profiles:jws:InappropriateSerializationSyntax

urn:oasis:names:tc:dss-x:2.0:profiles:jws:HeaderNameCollision

urn:oasis:names:tc:dss-x:2.0:profiles:jws:UnprotectedHeaderNotSupported

urn:oasis:names:tc:dss-x:2.0:profiles:jws:UnprotectedHeaderNameCollision

1. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

1. Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Editor** | **Changes Made** |
|  |  |  |  |