Profile for comprehensive multi-signature verification reports 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:

Juan Carlos Cruellas (cruellas@ac.upc.edu), [Univ Politecnica de Cataluna](http://www.ac.upc.edu/)

Stefan Hagen (stefan@hagen.link), Individual

Editor:

Detlef Hühnlein, *Federal Office for Information Security, Germany* <detlef.huehnlein@ecsec.de>

Andreas Kühne, (kuehne@trustable.de), 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 replaces or supersedes:

* Stefan Drees et al., Digital Signature Service Core Protocols, Elements, and Bindings, Version 1.0, OASIS Standard, 11 April 2007,
<http://docs.oasis-open.org/dss/v1.0/oasis-dss-core-spec-v1.0-os.pdf>

This specification is related to:

* Related specifications (hyperlink, if available)

Declared XML namespaces:

* urn:oasis:names:tc:dss-x:1.0:profiles:verificationreport:[schema](http://www.oasis-open.org/committees/download.php/33059/VerificationReport-CD1.xsd)#

Abstract:

This document defines a protocol and processing profile of the DSS Verifying Protocol specified in **[DSSCore]**, which allows to return individual signature verification reports for each signature in a verification request and include detailed information of the different steps taken during verification.

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 9](#_Toc482893695)

[1.1 Organization of DSS Core Protocols, Elements, and Bindings 9](#_Toc482893696)

[1.2 Terminology 9](#_Toc482893697)

[1.2.1 Terms and Definitions 9](#_Toc482893698)

[1.2.2 Abbreviated Terms 9](#_Toc482893699)

[1.3 Normative References 9](#_Toc482893700)

[1.4 Non-Normative References 11](#_Toc482893701)

[1.5 Typographical Conventions 11](#_Toc482893702)

[2 Design Considerations 12](#_Toc482893703)

[2.1 Construction Principles 12](#_Toc482893704)

[2.2 Domain Models 12](#_Toc482893705)

[2.2.1 Date and Time Model 12](#_Toc482893706)

[2.3 Schema Organization and Namespaces 12](#_Toc482893707)

[2.4 DSS Overview (Non-normative) 13](#_Toc482893708)

[2.5 Version 2.0 motivation [non-normative] 14](#_Toc482893709)

[2.6 Syntax variants 14](#_Toc482893710)

[3 Structure Models 15](#_Toc482893711)

[3.1 Type Base64DataType 15](#_Toc482893712)

[3.2 Type AnyType 16](#_Toc482893713)

[3.3 Type InternationalStringType 16](#_Toc482893714)

[3.4 Type KeyInfoType 17](#_Toc482893715)

[3.5 Element InputDocuments 18](#_Toc482893716)

[3.5.1 Type DocumentBaseType 19](#_Toc482893717)

[3.5.2 Type DocumentType 19](#_Toc482893718)

[3.5.2.1 XML Syntax 19](#_Toc482893719)

[3.5.2.2 JSON Syntax 20](#_Toc482893720)

[3.5.3 Type TransformedDataType 20](#_Toc482893721)

[3.5.3.1 XML Syntax 21](#_Toc482893722)

[3.5.3.2 JSON Syntax 21](#_Toc482893723)

[3.5.4 Type DocumentHashType 21](#_Toc482893724)

[3.5.4.1 XML Syntax 22](#_Toc482893725)

[3.5.4.2 JSON Syntax 22](#_Toc482893726)

[3.6 Element SignatureObject 22](#_Toc482893727)

[3.6.1 XML Syntax 23](#_Toc482893728)

[3.6.2 JSON Syntax 24](#_Toc482893729)

[3.7 Element Result 24](#_Toc482893730)

[3.7.1 XML Syntax 26](#_Toc482893731)

[3.7.2 JSON Syntax 26](#_Toc482893732)

[3.8 Common Optional Inputs 27](#_Toc482893733)

[3.8.1 Optional Input ServicePolicy 27](#_Toc482893734)

[3.8.1.1 XML Syntax 27](#_Toc482893735)

[3.8.1.2 JSON Syntax 27](#_Toc482893736)

[3.8.2 Optional Input ClaimedIdentity 27](#_Toc482893737)

[3.8.2.1 XML Syntax 28](#_Toc482893738)

[3.8.2.2 JSON Syntax 28](#_Toc482893739)

[3.8.3 Optional Input Language 28](#_Toc482893740)

[3.8.3.1 XML Syntax 28](#_Toc482893741)

[3.8.3.2 JSON Syntax 28](#_Toc482893742)

[3.8.4 Optional Input Profile 29](#_Toc482893743)

[3.8.4.1 XML Syntax 29](#_Toc482893744)

[3.8.4.2 JSON Syntax 29](#_Toc482893745)

[3.8.5 Optional Input Schemas 29](#_Toc482893746)

[3.8.5.1 XML Syntax 30](#_Toc482893747)

[3.8.5.2 JSON Syntax 30](#_Toc482893748)

[3.8.6 Type Optional Input ReturnTransformedDocument and Output TransformedDocument 30](#_Toc482893749)

[3.8.6.1 XML Syntax 31](#_Toc482893750)

[3.8.6.2 JSON Syntax 31](#_Toc482893751)

[3.9 OptionalInputsBaseType 31](#_Toc482893752)

[3.9.1.1 XML Syntax 32](#_Toc482893753)

[3.9.1.2 JSON Syntax 32](#_Toc482893754)

[3.10 Common Optional Outputs 33](#_Toc482893755)

[3.10.1 Optional Output Schemas 33](#_Toc482893756)

[3.11 OptionalOutputsBaseType 33](#_Toc482893757)

[3.11.1.1 XML Syntax 33](#_Toc482893758)

[3.11.1.2 JSON Syntax 34](#_Toc482893759)

[3.12 Type RequestBaseType 34](#_Toc482893760)

[3.12.1 XML Syntax 35](#_Toc482893761)

[3.12.2 JSON Syntax 35](#_Toc482893762)

[3.13 Type ResponseBaseType 35](#_Toc482893763)

[3.13.1 XML Syntax 35](#_Toc482893764)

[3.13.2 JSON Syntax 35](#_Toc482893765)

[4 The DSS Signing Protocol 37](#_Toc482893766)

[4.1 Element SignRequest 37](#_Toc482893767)

[4.1.1 XML Syntax 37](#_Toc482893768)

[4.1.2 JSON Syntax 37](#_Toc482893769)

[4.2 Element SignResponse 37](#_Toc482893770)

[4.2.1 XML Syntax 38](#_Toc482893771)

[4.2.2 JSON Syntax 38](#_Toc482893772)

[4.3 Processing for XML Signatures 39](#_Toc482893773)

[4.3.1 Basic Process for XML 39](#_Toc482893774)

[4.3.2 Process Variant for TransformedData 40](#_Toc482893775)

[4.3.3 Process Variant for DocumentHash 40](#_Toc482893776)

[4.4 Basic Processing for CMS Signatures 41](#_Toc482893777)

[4.4.1 Process Variant for DocumentHash 41](#_Toc482893778)

[4.5 Optional Inputs and Outputs 41](#_Toc482893779)

[4.5.1 Optional Input SignatureType 42](#_Toc482893780)

[4.5.1.1 XML Syntax 42](#_Toc482893781)

[4.5.1.2 JSON Syntax 42](#_Toc482893782)

[4.5.2 Optional Input AddTimestamp 42](#_Toc482893783)

[4.5.2.1 XML Syntax 42](#_Toc482893784)

[4.5.2.2 JSON Syntax 42](#_Toc482893785)

[4.5.2.3 Processing of signatures time-stamping 43](#_Toc482893786)

[4.5.2.3.1 Processing for CMS signatures time-stamping 43](#_Toc482893787)

[4.5.2.3.2 Processing for XML Timestamps on XML signatures 43](#_Toc482893788)

[4.5.2.4 Processing for RFC 3161 Timestamps on XML signatures 44](#_Toc482893789)

[4.5.3 Optional Input IntendedAudience 44](#_Toc482893790)

[4.5.3.1 XML Syntax 44](#_Toc482893791)

[4.5.3.2 JSON Syntax 44](#_Toc482893792)

[4.5.4 Optional Input KeySelector 45](#_Toc482893793)

[4.5.4.1 XML Syntax 45](#_Toc482893794)

[4.5.4.2 JSON Syntax 45](#_Toc482893795)

[4.5.5 Optional Input Properties 45](#_Toc482893796)

[4.5.5.1 XML Syntax 45](#_Toc482893797)

[4.5.5.2 JSON Syntax 46](#_Toc482893798)

[4.5.6 Optional Input IncludeObject 46](#_Toc482893799)

[4.5.6.1 XML Syntax 47](#_Toc482893800)

[4.5.6.2 JSON Syntax 47](#_Toc482893801)

[4.5.6.3 XML Signatures Variant Optional Input IncludeObject 47](#_Toc482893802)

[4.5.7 Optional Input IncludeEContent 48](#_Toc482893803)

[4.5.8 Enveloped Signatures, Optional Input SignaturePlacement and Output DocumentWithSignature 49](#_Toc482893804)

[4.5.8.1 XML Syntax 50](#_Toc482893805)

[4.5.8.2 JSON Syntax 51](#_Toc482893806)

[4.5.9 Optional Input SignedReferences 51](#_Toc482893807)

[4.5.9.1 XML Syntax 53](#_Toc482893808)

[4.5.9.2 JSON Syntax 53](#_Toc482893809)

[4.6 OptionalInputsSignType 53](#_Toc482893810)

[4.6.1.1 XML Syntax 54](#_Toc482893811)

[4.6.1.2 JSON Syntax 54](#_Toc482893812)

[4.7 OptionalOutputsSignType 55](#_Toc482893813)

[4.7.1.1 XML Syntax 55](#_Toc482893814)

[4.7.1.2 JSON Syntax 55](#_Toc482893815)

[5 The DSS Verifying Protocol 57](#_Toc482893816)

[5.1 Element VerifyRequest 57](#_Toc482893817)

[5.1.1 XML Syntax 57](#_Toc482893818)

[5.1.2 JSON Syntax 57](#_Toc482893819)

[5.2 Element VerifyResponse 58](#_Toc482893820)

[5.2.1 JSON Syntax 58](#_Toc482893821)

[5.3 Basic Processing for XML Signatures 58](#_Toc482893822)

[5.3.1 Multi-Signature Verification 59](#_Toc482893823)

[5.3.2 Signature Timestamp verification procedure 60](#_Toc482893824)

[5.3.2.1 Processing for RFC 3161 Timestamp tokens on CMS Signatures. 60](#_Toc482893825)

[5.3.2.2 Processing for XML timestamp tokens on XML signatures 60](#_Toc482893826)

[5.3.2.3 Processing for RFC 3161 timestamp tokens on XML Signatures 61](#_Toc482893827)

[5.4 Basic Processing for CMS Signatures 62](#_Toc482893828)

[5.5 Optional Inputs and Outputs 62](#_Toc482893829)

[5.5.1 Optional Input VerifyManifests and Output VerifyManifestResults 62](#_Toc482893830)

[5.5.1.1 XML Syntax 62](#_Toc482893831)

[5.5.1.2 JSON Syntax 63](#_Toc482893832)

[5.5.2 Optional Input UseVerificationTime 63](#_Toc482893833)

[5.5.2.1 XML Syntax 63](#_Toc482893834)

[5.5.2.2 JSON Syntax 64](#_Toc482893835)

[5.5.3 Optional Input/Output ReturnVerificationTimeInfo / VerificationTimeInfo 64](#_Toc482893836)

[5.5.3.1 XML Syntax 65](#_Toc482893837)

[5.5.3.2 JSON Syntax 65](#_Toc482893838)

[5.5.4 Optional Input AdditionalKeyInfo 66](#_Toc482893839)

[5.5.4.1 XML Syntax 66](#_Toc482893840)

[5.5.4.2 JSON Syntax 66](#_Toc482893841)

[5.5.5 Optional Input ReturnProcessingDetails and Output ProcessingDetails 66](#_Toc482893842)

[5.5.5.1 XML Syntax 67](#_Toc482893843)

[5.5.5.2 JSON Syntax 68](#_Toc482893844)

[5.5.6 Optional Input ReturnSigningTimeInfo and Output SigningTimeInfo 68](#_Toc482893845)

[5.5.6.1 XML Syntax 69](#_Toc482893846)

[5.5.6.2 JSON Syntax 69](#_Toc482893847)

[5.5.7 Optional Input ReturnSignerIdentity and Output SignerIdentity 70](#_Toc482893848)

[5.5.7.1 XML Syntax 70](#_Toc482893849)

[5.5.7.2 JSON Syntax 70](#_Toc482893850)

[5.5.8 Optional Input ReturnUpdatedSignature and Outputs DocumentWithSignature, UpdatedSignature 70](#_Toc482893851)

[5.5.8.1 XML Syntax 71](#_Toc482893852)

[5.5.8.2 JSON Syntax 72](#_Toc482893853)

[5.5.9 Optional Input ReturnTransformedDocument and Output TransformedDocument 72](#_Toc482893854)

[5.5.9.1 XML Syntax 72](#_Toc482893855)

[5.5.9.2 JSON Syntax 73](#_Toc482893856)

[5.5.10 Optional Input ReturnTimestampedSignature and Outputs DocumentWithSignature, TimestampedSignature 73](#_Toc482893857)

[5.5.10.1 XML Syntax 74](#_Toc482893858)

[5.5.10.2 JSON Syntax 74](#_Toc482893859)

[5.6 OptionalInputsVerifyType 74](#_Toc482893860)

[5.6.1.1 XML Syntax 75](#_Toc482893861)

[5.6.1.2 JSON Syntax 75](#_Toc482893862)

[5.7 OptionalOutputsVerifyType 76](#_Toc482893863)

[5.7.1.1 XML Syntax 77](#_Toc482893864)

[5.7.1.2 JSON Syntax 77](#_Toc482893865)

[6 DSS Core Elements 78](#_Toc482893866)

[6.1 Element Timestamp 78](#_Toc482893867)

[6.1.1 XML Timestamp Token 78](#_Toc482893868)

[6.1.2 Element TstInfo 79](#_Toc482893869)

[6.1.2.1 XML Syntax 79](#_Toc482893870)

[6.1.2.2 JSON Syntax 79](#_Toc482893871)

[6.2 Element RequesterIdentity 80](#_Toc482893872)

[6.2.1.1 XML Syntax 80](#_Toc482893873)

[6.2.1.2 JSON Syntax 80](#_Toc482893874)

[7 DSS Core Bindings 81](#_Toc482893875)

[7.1 HTTP POST Transport Binding 81](#_Toc482893876)

[7.2 SOAP 1.2 Transport Binding 81](#_Toc482893877)

[7.2.1 SOAP Attachment Feature and Element <AttachmentReference> 82](#_Toc482893878)

[7.2.1.1 Signing Protocol, Processing for XML Signatures, Process Variant for <AttachmentReference> 82](#_Toc482893879)

[7.2.1.2 Verifying Protocol, Processing for XML Signatures, Process Variant for <AttachmentReference> 83](#_Toc482893880)

[7.2.1.3 Signing Protocol, Basic Processing for CMS Signatures, Process Variant for <AttachmentReference> 83](#_Toc482893881)

[7.2.1.4 Verifying Protocol, Basic Processing for CMS Signatures, Process Variant for <AttachmentReference> 83](#_Toc482893882)

[8 Processing Model 84](#_Toc482893883)

[9 JSON Format 85](#_Toc482893884)

[9.1 JSON – Type Base64DataType 85](#_Toc482893885)

[9.2 JSON – Type AnyType 85](#_Toc482893886)

[9.3 JSON – Type InternationalStringType 86](#_Toc482893887)

[9.4 JSON – Type KeyInfoType 86](#_Toc482893888)

[9.5 JSON – Element InputDocuments 87](#_Toc482893889)

[9.5.1 JSON – Type DocumentBaseType 87](#_Toc482893890)

[10 XML Format 89](#_Toc482893891)

[10.1 XML – Type Base64DataType 89](#_Toc482893892)

[10.2 XML – Type AnyType 89](#_Toc482893893)

[10.3 XML – Type InternationalStringType 90](#_Toc482893894)

[10.4 XML – Type KeyInfoType 90](#_Toc482893895)

[10.5 XML – Element InputDocuments 91](#_Toc482893896)

[10.5.1 XML – Type DocumentBaseType 92](#_Toc482893897)

[10.6 AnElement – REMOVE\_ME\_AFTER\_FIRST\_PASS 93](#_Toc482893898)

[11 DSS-Defined Identifiers 95](#_Toc482893899)

[11.1 Signature Type Identifiers 95](#_Toc482893900)

[11.1.1 XML Signature 95](#_Toc482893901)

[11.1.2 XML TimeStampToken 95](#_Toc482893902)

[11.1.3 RFC 3161 TimeStampToken 95](#_Toc482893903)

[11.1.4 CMS Signature 95](#_Toc482893904)

[11.1.5 PGP Signature 95](#_Toc482893905)

[12 Conformance 96](#_Toc482893906)

[12.1 Conformance as a DSS version 2.0 document 96](#_Toc482893907)

[12.1.1 Conformance for XML format 96](#_Toc482893908)

[12.1.2 Conformance for JSON format 96](#_Toc482893909)

[Appendix A. Acknowledgments 97](#_Toc482893910)

[Appendix B. 98](#_Toc482893911)

[B.1 Use of Exclusive Canonicalization 98](#_Toc482893912)

[B.2 More Complex Response Example 98](#_Toc482893913)

[Appendix C. 99](#_Toc482893914)

[C.1 Element InputDocuments 99](#_Toc482893915)

[C.1.1 XML Syntax 100](#_Toc482893916)

[C.1.2 JSON Syntax 100](#_Toc482893917)

[C.1.3 Type TransformedDataType 101](#_Toc482893918)

[Appendix D. Table of Types, Elements and Attributes 103](#_Toc482893919)

[Appendix E. List of Figures 105](#_Toc482893920)

[Appendix F. Index 106](#_Toc482893921)

[Appendix G. JSON Helpers 107](#_Toc482893922)

[Appendix H. Revision History 108](#_Toc482893923)

# Introduction

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

This document defines a protocol and processing profile of the DSS Verifying Protocol specified in **[DSSCore]**, which allows to support the verification of multiple signatures within a VerifyRequest component and include detailed information of the different steps taken during verification.

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 Schema*. OASIS, ToDo.

**[DSSVR-XSD]** D. Hühnlein, A. Kuehne, et. al.: “*DSS Verification**Report 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>.

**[SAMLCore1.1]** E. Maler et al. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V 1.1. OASIS, November 2002.
<http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf>

**[SAMLCore2.0]** “*Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0”,* S. Cantor, J. Kemp, R. Philpott, E. Maler, Editors, OASIS Standard, 15 March 2005.

**[XAdES]** ETSI: “*XML Advanced Electronic Signatures (XAdES)”*, ETSI TS 101 903, Version 1.3.2, March 2006

 **[XMLDSIG]** *D. Eastlake et al. XML-Signature Syntax and Processing. W3C Recommendation, February 2002.*<http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/>

[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.

# Design Considerations

Blurb

## Construction Principles

## Domain Models

### Date and Time Model

The specific concept of date and time used in this document is defined in this section and noted in subsequent usage as**:**

DateTime

« All date time values inside a DSS document MUST adhere to the ISO 8601 [[ISO8601](#refISO8601)] basic or extended Format (as given there in section 4.3.2 “Complete representations” and with the addition of decimal fractions for seconds, similar to ibid. section 4.2.2.4 “Representations with decimal fraction” but with the full stop (.) being the preferred separator for DSS). » [DSS-2.2.1-1].

## Schema Organization and Namespaces

The structures described in this specification are contained in the schema file **[Core2.0-XSD]**. All schema listings in the current document are excerpts from the schema file. In the case of a disagreement between the schema file and this document, the schema file takes precedence.

This schema is associated with the following XML namespace:

urn:oasis:names:tc:dss:2.0:core:schema

If a future version of this specification is needed, it will use a different namespace.

Conventional XML namespace prefixes are used in the schema:

* The prefix dss2: stands for the DSS core namespace **[DSS2XSD]**.
* The prefix xs: stands for the W3C XML Schema namespace **[Schema1]**.
* The prefix vr: (or no prefix) stands for this profiles namespace **[DSSVR-XSD]**.
* The prefix ds: stands for the W3C XML Signature namespace **[XMLSig]**.
* The prefix saml: stands for the OASIS SAML Schema namespace **[SAMLCore1.1]**.
* The prefix saml2: stands for the OASIS SAML Schema namespace.
* The prefix xades: stands for ETSI XML Advanced Electronic Signatures (XAdES) document **[XAdES]**.

Applications MAY use different namespace prefixes, and MAY use whatever namespace defaulting/scoping conventions they desire, as long as they are compliant with the Namespaces in XML specification **[XML-ns]**.

The following schema fragment defines the XML namespaces and other header information for the DSS core schema:

<xs:schema xmlns:dss2="urn:oasis:names:tc:dss:2.0:core:schema"

 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"

 xmlns:xs="http://www.w3.org/2001/XMLSchema"

 xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"

 targetNamespace="urn:oasis:names:tc:dss:2.0:core:schema"

 elementFormDefault="qualified"

 attributeFormDefault="unqualified">

<xs:annotation>

 <xs:documentation xml:lang="en">This Schema defines the Digital Signature Service Core Protocols, Elements, and Bindings Committee Draft 1 for Public Review</xs:documentation>

</xs:annotation>

<xs:import namespace="http://www.w3.org/2000/09/xmldsig#" schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>

<xs:import namespace="urn:oasis:names:tc:SAML:1.0:assertion" schemaLocation="http://www.oasis-open.org/committees/download.php/3408/oasis-sstc-saml-schema-protocol-1.1.xsd"/>

<xs:import namespace="http://www.w3.org/XML/1998/namespace" schemaLocation="http://www.w3.org/2001/xml.xsd"/>

## DSS Overview (Non-normative)

While the DSS Verifying Protocol specified in **[DSSCore]** allows to verify digital signatures and time stamps, this protocol is fairly limited with respect to the verification of multiple signatures in a single request (cf. Section 4.3.1 of **[DSSCore]**).

In a similar manner it is possible to request and provide processing details (cf. Section 4.5.5 of **[DSSCore]**), but this simple mechanism does not support the verification of multiple signatures in a single request. And there are no defined structures yet, which reflect the necessary steps in the verification of a complex signature, like an advanced electronic signature according to the European Directive **[EC/1999/93]** for example.

Therefore, the present profile defines how

* individual verification results may be returned, if multiple signatures are part of a VerifyRequest component and
* detailed information gathered in the various steps taken during verification may be included in the response to form a comprehensive verification report.

The requester MAY request the activation of this profile by sending a ReturnVerificationReport element (cf. Section 3.1) in OptionalInputs component. A responder, which conforms to the present profile SHALL return a VerificationReport element (cf. Section 3.2) in OptionalOutputs component.

## Syntax variants

This version of the DSS/X profile document handles the representation of requests and response elements according to the JSON and XML syntax. The general semantics of the elements is discussed in the element’s main section. Details of the JSON or XML formats are discussed in specific subsections

* JSON syntax
* XML syntax

# Structure Models

## Structure Models defined in this document

The XML elements of this section are defined in the XML namespace 'http://ws.openecard.org/chipgateway'.

[namespace http://ws.openecard.org/chipgateway explanation]

### Component CertificateFilter

#### Semantics

[component CertificateFilter normative details]

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

* The optional Policy element MUST contain a string. [sub component Policy details]
* The optional Issuer element MUST contain a string. [sub component Issuer details]
* The optional KeyUsage element MUST contain a sub-component. A given element MUST satisfy the requirements specified in section KeyUsage. [sub component KeyUsage details]

Non-normative Comment:

[component CertificateFilter non normative details]

#### XML Syntax

The XML type CertificateFilterType SHALL implement the requirements defined in the CertificateFilter component.

The CertificateFilterType 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.

<complexType name="CertificateFilterType">

 <sequence>

 <element maxOccurs="1" minOccurs="0" name="Policy" type="string"/>

 <element maxOccurs="1" minOccurs="0" name="Issuer" type="string"/>

 <element maxOccurs="1" minOccurs="0" name="KeyUsage" type="cg:KeyUsageType"/>

 </sequence>

</complexType>

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

#### JSON Syntax

The CertificateFilterType JSON object SHALL implement in JSON syntax the requirements defined in the CertificateFilter component.

Properties of the JSON object SHALL implement the sub-components of CertificateFilter using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| Policy | policy | [] |
| Issuer | iss | [] |
| KeyUsage | usage | [] |

The CertificateFilterType JSON schema listed below for information.

"cg-CertificateFilterType": {

 "$xsd-full-type": "cg:CertificateFilterType",

 "type": "object",

 "properties": {

 "policy": {

 "type": "string"

 },

 "iss": {

 "type": "string"

 },

 "usage": {

 "type": "string"

 }

 }

}

[component CertificateFilter JSON schema details]

### Component CertificateInfo

#### Semantics

[component CertificateInfo normative details]

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

* The DIDName element MUST contain one instance of a sub-component. This element MUST satisfy the requirements specified in section Name. [sub component DIDName details]
* The Algorithm element MUST contain one instance of a string. [sub component Algorithm details]
* The Certificate element MUST occur 1 or more times containing base64 encoded binary data. [sub component Certificate details]
* The UniqueSSN element MUST contain one instance of a string. [sub component UniqueSSN details]

Non-normative Comment:

[component CertificateInfo non normative details]

#### XML Syntax

The XML type CertificateInfoType SHALL implement the requirements defined in the CertificateInfo component.

The CertificateInfoType 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.

<complexType name="CertificateInfoType">

 <sequence>

 <element name="DIDName" type="cg:NameType"/>

 <element name="Algorithm" type="string"/>

 <element maxOccurs="unbounded" minOccurs="1" name="Certificate" type="base64Binary"/>

 <element name="UniqueSSN" type="string"/>

 </sequence>

</complexType>

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

#### JSON Syntax

The CertificateInfoType JSON object SHALL implement in JSON syntax the requirements defined in the CertificateInfo component.

Properties of the JSON object SHALL implement the sub-components of CertificateInfo using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| DIDName | did | [] |
| Algorithm | alg | [] |
| Certificate | cert | [] |
| UniqueSSN | ssn | [] |

The CertificateInfoType JSON schema listed below for information.

"cg-CertificateInfoType": {

 "$xsd-full-type": "cg:CertificateInfoType",

 "type": "object",

 "properties": {

 "didname": {

 "type": "string"

 },

 "did": {

 "type": "string"

 },

 "alg": {

 "type": "string"

 },

 "cert": {

 "type": "array",

 "items": {

 "type": "string"

 }

 },

 "ssn": {

 "type": "string"

 }

 },

 "required": ["did", "alg", "cert", "ssn"]

}

[component CertificateInfo JSON schema details]

### Component ConnectionHandle

#### Semantics

[component ConnectionHandle normative details]

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

* The CardType element MUST contain one instance of a URI. [sub component CardType details]
* The optional SlotHandle element MUST contain hex encoded binary data. [sub component SlotHandle details]

Non-normative Comment:

[component ConnectionHandle non normative details]

#### XML Syntax

The XML type ConnectionHandleType SHALL implement the requirements defined in the ConnectionHandle component.

The ConnectionHandleType 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.

<complexType name="ConnectionHandleType">

 <sequence>

 <element name="CardType" type="anyURI"/>

 <element maxOccurs="1" minOccurs="0" name="SlotHandle" type="hexBinary"/>

 </sequence>

</complexType>

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

#### JSON Syntax

The ConnectionHandleType JSON object SHALL implement in JSON syntax the requirements defined in the ConnectionHandle component.

Properties of the JSON object SHALL implement the sub-components of ConnectionHandle using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| CardType | cardType | [] |
| SlotHandle | slot | [] |

The ConnectionHandleType JSON schema listed below for information.

"cg-ConnectionHandleType": {

 "$xsd-full-type": "cg:ConnectionHandleType",

 "type": "object",

 "properties": {

 "cardType": {

 "type": "string"

 },

 "slot": {

 "type": "array",

 "items": {

 "type": "integer"

 }

 }

 },

 "required": ["cardType"]

}

[component ConnectionHandle JSON schema details]

### Component KeyUsage

#### Semantics

[component KeyUsage normative details]

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

* The value element MUST contain one instance of a string. Its value is limited to an item of the following set:
AUTHENTICATION
SIGNATURE
ENCRYPTION
[sub component value details]

Non-normative Comment:

[component KeyUsage non normative details]

#### XML Syntax

The XML type KeyUsageType SHALL implement the requirements defined in the KeyUsage component.

The KeyUsageType 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.

<simpleType name="KeyUsageType">

 <restriction base="string">

 <enumeration value="AUTHENTICATION"/>

 <enumeration value="SIGNATURE"/>

 <enumeration value="ENCRYPTION"/>

 </restriction>

</simpleType>

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

#### JSON Syntax

The KeyUsageType JSON object SHALL implement in JSON syntax the requirements defined in the KeyUsage component.

Properties of the JSON object SHALL implement the sub-components of KeyUsage using JSON-specific names mapped as shown in the table below.

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

The KeyUsageType JSON schema listed below for information.

"cg-KeyUsageType": {

 "$xsd-full-type": "cg:KeyUsageType",

 "type": "string",

 "enum": ["AUTHENTICATION", "SIGNATURE", "ENCRYPTION"]

}

[component KeyUsage JSON schema details]

### Component Name

#### Semantics

[component Name normative details]

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

* The value element MUST contain one instance of a string. The length is limited to 1 ... 255 characters. [sub component value details]

Non-normative Comment:

[component Name non normative details]

#### XML Syntax

The XML type NameType SHALL implement the requirements defined in the Name component.

The NameType 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.

<simpleType name="NameType">

 <restriction base="normalizedString">

 <minLength value="1"/>

 <maxLength value="255"/>

 <whiteSpace value="collapse"/>

 </restriction>

</simpleType>

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

#### JSON Syntax

The NameType JSON object SHALL implement in JSON syntax the requirements defined in the Name component.

Properties of the JSON object SHALL implement the sub-components of Name using JSON-specific names mapped as shown in the table below.

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

The NameType JSON schema listed below for information.

"cg-NameType": {

 "$xsd-full-type": "cg:NameType",

 "type": "string",

 "minLength": "1",

 "maxLength": "255"

}

[component Name JSON schema details]

### Component Response

#### Semantics

[component Response normative details]

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

* The Result element MUST contain one instance of a sub-component. This element MUST satisfy the requirements specified in section Result. [sub component Result details]

Non-normative Comment:

[component Response non normative details]

#### XML Syntax

The XML type ResponseType SHALL implement the requirements defined in the Response component.

The ResponseType 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.

<complexType name="ResponseType">

 <sequence>

 <element name="Result" type="dsb:ResultType"/>

 </sequence>

</complexType>

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

#### JSON Syntax

The ResponseType JSON object SHALL implement in JSON syntax the requirements defined in the Response component.

Properties of the JSON object SHALL implement the sub-components of Response using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| Result | result | [] |

The ResponseType JSON schema listed below for information.

"cg-ResponseType": {

 "$xsd-full-type": "cg:ResponseType",

 "type": "object",

 "properties": {

 "result": {

 "$ref": "#/definitions/dsb-ResultType"

 }

 },

 "required": ["result"]

}

[component Response JSON schema details]

### Component TokenInfo

#### Semantics

[component TokenInfo normative details]

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

* The optional ConnectionHandle element MUST contain a sub-component. A given element MUST satisfy the requirements specified in section ConnectionHandle. [sub component ConnectionHandle details]
* The optional HasProtectedAuthPath element MUST contain a boolean. [sub component HasProtectedAuthPath details]
* The optional NeedsPinForCertAccess element MUST contain a boolean. [sub component NeedsPinForCertAccess details]
* The optional NeedsPinForPrivateKeyAccess element MUST contain a boolean. [sub component NeedsPinForPrivateKeyAccess details]
* The optional Algorithm element MAY occur zero or more times containing a string. [sub component Algorithm details]

Non-normative Comment:

[component TokenInfo non normative details]

#### XML Syntax

The XML type TokenInfoType SHALL implement the requirements defined in the TokenInfo component.

The TokenInfoType 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.

<complexType name="TokenInfoType">

 <sequence>

 <element maxOccurs="1" minOccurs="0" ref="cg:ConnectionHandle"/>

 <element maxOccurs="1" minOccurs="0" name="HasProtectedAuthPath" type="boolean"/>

 <element maxOccurs="1" minOccurs="0" name="NeedsPinForCertAccess" type="boolean"/>

 <element maxOccurs="1" minOccurs="0" name="NeedsPinForPrivateKeyAccess" type="boolean"/>

 <element maxOccurs="unbounded" minOccurs="0" name="Algorithm" type="string"/>

 </sequence>

</complexType>

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

#### JSON Syntax

The TokenInfoType JSON object SHALL implement in JSON syntax the requirements defined in the TokenInfo component.

Properties of the JSON object SHALL implement the sub-components of TokenInfo using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| ConnectionHandle | conn | [] |
| HasProtectedAuthPath | protAuthPath | [] |
| NeedsPinForCertAccess | certNeedsPin | [] |
| NeedsPinForPrivateKeyAccess | privNeedsPin | [] |
| Algorithm | alg | [] |

The TokenInfoType JSON schema listed below for information.

"cg-TokenInfoType": {

 "$xsd-full-type": "cg:TokenInfoType",

 "type": "object",

 "properties": {

 "conn": {

 "$ref": "#/definitions/cg-ConnectionHandleType"

 },

 "protAuthPath": {

 "type": "boolean"

 },

 "certNeedsPin": {

 "type": "boolean"

 },

 "privNeedsPin": {

 "type": "boolean"

 },

 "alg": {

 "type": "array",

 "items": {

 "type": "string"

 }

 }

 }

}

[component TokenInfo JSON schema details]

### Component HelloRequest

#### Semantics

[component HelloRequest normative details]

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

* The Challenge element MUST contain one instance of hex encoded binary data. [sub component Challenge details]
* The Version element MUST contain one instance of a string. [sub component Version details]
* The SessionIdentifier element MUST contain one instance of a string. [sub component SessionIdentifier details]

Non-normative Comment:

[component HelloRequest non normative details]

#### XML Syntax

The XML type HelloRequestType SHALL implement the requirements defined in the HelloRequest component.

The HelloRequestType 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.

<complexType name="HelloRequestType">

 <sequence>

 <element name="Challenge" type="hexBinary"/>

 <element name="Version" type="string"/>

 <element name="SessionIdentifier" type="string"/>

 </sequence>

</complexType>

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

#### JSON Syntax

The HelloRequestType JSON object SHALL implement in JSON syntax the requirements defined in the HelloRequest component.

Properties of the JSON object SHALL implement the sub-components of HelloRequest using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| Challenge | chall | [] |
| Version | ver | [] |
| SessionIdentifier | sessId | [] |

The HelloRequestType JSON schema listed below for information.

"cg-HelloRequestType": {

 "$xsd-full-type": "cg:HelloRequestType",

 "type": "object",

 "properties": {

 "chall": {

 "type": "array",

 "items": {

 "type": "integer"

 }

 },

 "ver": {

 "type": "string"

 },

 "sessId": {

 "type": "string"

 }

 },

 "required": ["chall", "ver", "sessId"]

}

[component HelloRequest JSON schema details]

### Component HelloResponse

#### Semantics

[component HelloResponse normative details]

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

* The Signature element MUST contain base64 encoded binary data. [sub component Signature details]
* The optional MinimumVersion element MUST contain a string. [sub component MinimumVersion details]
* The optional DownloadAddress element MUST contain a URI. [sub component DownloadAddress details]
* The optional WebOrigin element MAY occur zero or more times containing a string. [sub component WebOrigin details]

A set of sub-components is inherited from component Response and is not repeated here.

Non-normative Comment:

[component HelloResponse non normative details]

#### XML Syntax

The XML type HelloResponseType SHALL implement the requirements defined in the HelloResponse component.

The HelloResponseType 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.

<complexType name="HelloResponseType">

 <complexContent>

 <extension base="cg:ResponseType">

 <sequence maxOccurs="1" minOccurs="0">

 <element name="Signature" type="base64Binary"/>

 <element maxOccurs="1" minOccurs="0" name="MinimumVersion" type="string"/>

 <element maxOccurs="1" minOccurs="0" name="DownloadAddress" type="anyURI"/>

 <element maxOccurs="unbounded" minOccurs="0" name="WebOrigin" type="string"/>

 </sequence>

 </extension>

 </complexContent>

</complexType>

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

#### JSON Syntax

The HelloResponseType JSON object SHALL implement in JSON syntax the requirements defined in the HelloResponse component.

Properties of the JSON object SHALL implement the sub-components of HelloResponse using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| Signature | sig | [] |
| MinimumVersion | minVer | [] |
| DownloadAddress | downloasdAddress | [] |
| WebOrigin | webOrigin | [] |

The HelloResponseType JSON schema listed below for information.

"cg-HelloResponseType": {

 "$xsd-full-type": "cg:HelloResponseType",

 "type": "object",

 "properties": {

 "result": {

 "$ref": "#/definitions/dsb-ResultType"

 },

 "sig": {

 "type": "array",

 "items": {

 "type": "integer"

 }

 },

 "minVer": {

 "type": "string"

 },

 "downloasdAddress": {

 "type": "string"

 },

 "webOrigin": {

 "type": "array",

 "items": {

 "type": "string"

 }

 }

 },

 "required": ["sig"]

}

[component HelloResponse JSON schema details]

### Component GetCommand

#### Semantics

[component GetCommand normative details]

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

* The SessionIdentifier element MUST contain one instance of a string. [sub component SessionIdentifier details]
* The optional TokenInfo element MAY occur zero or more times containing a sub-component. If present each instance MUST satisfy the requirements specified in section TokenInfo. [sub component TokenInfo details]

Non-normative Comment:

[component GetCommand non normative details]

#### XML Syntax

The XML type GetCommandType SHALL implement the requirements defined in the GetCommand component.

The GetCommandType 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.

<complexType name="GetCommandType">

 <sequence>

 <element name="SessionIdentifier" type="string"/>

 <element maxOccurs="unbounded" minOccurs="0" ref="cg:TokenInfo"/>

 </sequence>

</complexType>

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

#### JSON Syntax

The GetCommandType JSON object SHALL implement in JSON syntax the requirements defined in the GetCommand component.

Properties of the JSON object SHALL implement the sub-components of GetCommand using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| SessionIdentifier |  | [] |
| TokenInfo | tokenInfo | [] |

The GetCommandType JSON schema listed below for information.

"cg-GetCommandType": {

 "$xsd-full-type": "cg:GetCommandType",

 "type": "object",

 "properties": {

 "sessionIdentifier": {

 "type": "string"

 },

 "tokenInfo": {

 "type": "array",

 "items": {

 "$ref": "#/definitions/cg-TokenInfoType"

 }

 }

 }

}

[component GetCommand JSON schema details]

### Component Command

#### Semantics

[component Command normative details]

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

* The optional ListTokensRequest element MUST contain one instance of a sub-component. This element MUST satisfy the requirements specified in section ListTokensRequest. [sub component ListTokensRequest details]
* The optional ListCertificatesRequest element MUST contain one instance of a sub-component. This element MUST satisfy the requirements specified in section ListCertificatesRequest. [sub component ListCertificatesRequest details]
* The optional SignRequest element MUST contain one instance of a sub-component. This element MUST satisfy the requirements specified in section SignRequest. [sub component SignRequest details]
* The optional Terminate element MUST contain one instance of a sub-component. This element MUST satisfy the requirements specified in section Terminate. [sub component Terminate details]

Non-normative Comment:

[component Command non normative details]

#### XML Syntax

The XML type CommandType SHALL implement the requirements defined in the Command component.

The CommandType 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.

<complexType name="CommandType">

 <choice>

 <element ref="cg:ListTokensRequest"/>

 <element ref="cg:ListCertificatesRequest"/>

 <element ref="cg:SignRequest"/>

 <element ref="cg:Terminate"/>

 </choice>

</complexType>

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

#### JSON Syntax

The CommandType JSON object SHALL implement in JSON syntax the requirements defined in the Command component.

Properties of the JSON object SHALL implement the sub-components of Command using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| ListTokensRequest | tokensReq | [] |
| ListCertificatesRequest | certsReq | [] |
| SignRequest | signReq | [] |
| Terminate | term | [] |

The CommandType JSON schema listed below for information.

"cg-CommandType": {

 "$xsd-full-type": "cg:CommandType",

 "type": "object",

 "properties": {

 "tokensReq": {

 "$ref": "#/definitions/cg-ListTokensRequestType"

 },

 "certsReq": {

 "$ref": "#/definitions/cg-ListCertificatesRequestType"

 },

 "signReq": {

 "$ref": "#/definitions/cg-SignRequestType"

 },

 "term": {

 "$ref": "#/definitions/cg-TerminateType"

 }

 },

 "minProperties": 1,

 "maxProperties": 1

}

[component Command JSON schema details]

### Component ListTokensRequest

#### Semantics

[component ListTokensRequest normative details]

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

* The MaxWaitSeconds element MUST contain one instance of a positive integer. [sub component MaxWaitSeconds details]
* The TokenInfo element MUST occur 1 or more times containing a sub-component. Each instance MUST satisfy the requirements specified in section TokenInfo. [sub component TokenInfo details]

Non-normative Comment:

[component ListTokensRequest non normative details]

#### XML Syntax

The XML type ListTokensRequestType SHALL implement the requirements defined in the ListTokensRequest component.

The ListTokensRequestType 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.

<complexType name="ListTokensRequestType">

 <sequence>

 <element name="MaxWaitSeconds" type="positiveInteger"/>

 <element maxOccurs="unbounded" minOccurs="1" name="TokenInfo" type="cg:TokenInfoType"/>

 </sequence>

</complexType>

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

#### JSON Syntax

The ListTokensRequestType JSON object SHALL implement in JSON syntax the requirements defined in the ListTokensRequest component.

Properties of the JSON object SHALL implement the sub-components of ListTokensRequest using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| MaxWaitSeconds | waitSecs | [] |
| TokenInfo | tokenInfo | [] |

The ListTokensRequestType JSON schema listed below for information.

"cg-ListTokensRequestType": {

 "$xsd-full-type": "cg:ListTokensRequestType",

 "type": "object",

 "properties": {

 "waitSecs": {

 "type": "integer"

 },

 "tokenInfo": {

 "type": "array",

 "items": {

 "$ref": "#/definitions/cg-TokenInfoType"

 }

 }

 },

 "required": ["waitSecs", "tokenInfo"]

}

[component ListTokensRequest JSON schema details]

### Component ListTokensResponse

#### Semantics

[component ListTokensResponse normative details]

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

* The SessionIdentifier element MUST contain one instance of a string. [sub component SessionIdentifier details]
* The optional TokenInfo element MAY occur zero or more times containing a sub-component. If present each instance MUST satisfy the requirements specified in section TokenInfo. [sub component TokenInfo details]

A set of sub-components is inherited from component Response and is not repeated here.

Non-normative Comment:

[component ListTokensResponse non normative details]

#### XML Syntax

The XML type ListTokensResponseType SHALL implement the requirements defined in the ListTokensResponse component.

The ListTokensResponseType 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.

<complexType name="ListTokensResponseType">

 <complexContent>

 <extension base="cg:ResponseType">

 <sequence>

 <element name="SessionIdentifier" type="string"/>

 <element maxOccurs="unbounded" minOccurs="0" name="TokenInfo" type="cg:TokenInfoType"/>

 </sequence>

 </extension>

 </complexContent>

</complexType>

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

#### JSON Syntax

The ListTokensResponseType JSON object SHALL implement in JSON syntax the requirements defined in the ListTokensResponse component.

Properties of the JSON object SHALL implement the sub-components of ListTokensResponse using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| SessionIdentifier | sessionId | [] |
| TokenInfo | tokenInfo | [] |

The ListTokensResponseType JSON schema listed below for information.

"cg-ListTokensResponseType": {

 "$xsd-full-type": "cg:ListTokensResponseType",

 "type": "object",

 "properties": {

 "result": {

 "$ref": "#/definitions/dsb-ResultType"

 },

 "sessionId": {

 "type": "string"

 },

 "tokenInfo": {

 "type": "array",

 "items": {

 "$ref": "#/definitions/cg-TokenInfoType"

 }

 }

 },

 "required": ["sessionId"]

}

[component ListTokensResponse JSON schema details]

### Component ListCertificatesRequest

#### Semantics

[component ListCertificatesRequest normative details]

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

* The MaxWaitSeconds element MUST contain one instance of a positive integer. [sub component MaxWaitSeconds details]
* The SlotHandle element MUST contain one instance of hex encoded binary data. [sub component SlotHandle details]
* The optional PIN element MUST contain a string. [sub component PIN details]
* The optional CertificateFilter element MAY occur zero or more times containing a sub-component. If present each instance MUST satisfy the requirements specified in section CertificateFilter. [sub component CertificateFilter details]

Non-normative Comment:

[component ListCertificatesRequest non normative details]

#### XML Syntax

The XML type ListCertificatesRequestType SHALL implement the requirements defined in the ListCertificatesRequest component.

The ListCertificatesRequestType 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.

<complexType name="ListCertificatesRequestType">

 <sequence>

 <element name="MaxWaitSeconds" type="positiveInteger"/>

 <element name="SlotHandle" type="hexBinary"/>

 <element maxOccurs="1" minOccurs="0" name="PIN" type="string"/>

 <element maxOccurs="unbounded" minOccurs="0" name="CertificateFilter" type="cg:CertificateFilterType"/>

 </sequence>

</complexType>

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

#### JSON Syntax

The ListCertificatesRequestType JSON object SHALL implement in JSON syntax the requirements defined in the ListCertificatesRequest component.

Properties of the JSON object SHALL implement the sub-components of ListCertificatesRequest using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| MaxWaitSeconds | waitSecs | [] |
| SlotHandle | slot | [] |
| PIN | pin | [] |
| CertificateFilter | certFilter | [] |

The ListCertificatesRequestType JSON schema listed below for information.

"cg-ListCertificatesRequestType": {

 "$xsd-full-type": "cg:ListCertificatesRequestType",

 "type": "object",

 "properties": {

 "waitSecs": {

 "type": "integer"

 },

 "slot": {

 "type": "array",

 "items": {

 "type": "integer"

 }

 },

 "pin": {

 "type": "string"

 },

 "certFilter": {

 "type": "array",

 "items": {

 "$ref": "#/definitions/cg-CertificateFilterType"

 }

 }

 },

 "required": ["waitSecs", "slot"]

}

[component ListCertificatesRequest JSON schema details]

### Component ListCertificatesResponse

#### Semantics

[component ListCertificatesResponse normative details]

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

* The SessionIdentifier element MUST contain one instance of a string. [sub component SessionIdentifier details]
* The optional RetryCounter element MUST contain a non-negative integer. [sub component RetryCounter details]
* The optional CertificateInfo element MAY occur zero or more times containing a sub-component. If present each instance MUST satisfy the requirements specified in section CertificateInfo. [sub component CertificateInfo details]

A set of sub-components is inherited from component Response and is not repeated here.

Non-normative Comment:

[component ListCertificatesResponse non normative details]

#### XML Syntax

The XML type ListCertificatesResponseType SHALL implement the requirements defined in the ListCertificatesResponse component.

The ListCertificatesResponseType 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.

<complexType name="ListCertificatesResponseType">

 <complexContent>

 <extension base="cg:ResponseType">

 <sequence maxOccurs="1" minOccurs="1">

 <element name="SessionIdentifier" type="string"/>

 <element maxOccurs="1" minOccurs="0" name="RetryCounter" type="nonNegativeInteger"/>

 <element maxOccurs="unbounded" minOccurs="0" ref="cg:CertificateInfo"/>

 </sequence>

 </extension>

 </complexContent>

</complexType>

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

#### JSON Syntax

The ListCertificatesResponseType JSON object SHALL implement in JSON syntax the requirements defined in the ListCertificatesResponse component.

Properties of the JSON object SHALL implement the sub-components of ListCertificatesResponse using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| SessionIdentifier | sessionId | [] |
| RetryCounter | retryCnt | [] |
| CertificateInfo | certInfo | [] |

The ListCertificatesResponseType JSON schema listed below for information.

"cg-ListCertificatesResponseType": {

 "$xsd-full-type": "cg:ListCertificatesResponseType",

 "type": "object",

 "properties": {

 "result": {

 "$ref": "#/definitions/dsb-ResultType"

 },

 "sessionId": {

 "type": "string"

 },

 "retryCnt": {

 "type": "integer"

 },

 "certInfo": {

 "type": "array",

 "items": {

 "$ref": "#/definitions/cg-CertificateInfoType"

 }

 }

 },

 "required": ["sessionId"]

}

[component ListCertificatesResponse JSON schema details]

### Component SignRequest

#### Semantics

[component SignRequest normative details]

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

* The MaxWaitSeconds element MUST contain one instance of a positive integer. [sub component MaxWaitSeconds details]
* The SlotHandle element MUST contain one instance of hex encoded binary data. [sub component SlotHandle details]
* The DIDName element MUST contain one instance of a sub-component. This element MUST satisfy the requirements specified in section Name. [sub component DIDName details]
* The optional PIN element MUST contain a string. [sub component PIN details]
* The Message element MUST contain one instance of hex encoded binary data. [sub component Message details]

Non-normative Comment:

[component SignRequest non normative details]

#### XML Syntax

The XML type SignRequestType SHALL implement the requirements defined in the SignRequest component.

The SignRequestType 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.

<complexType name="SignRequestType">

 <sequence>

 <element name="MaxWaitSeconds" type="positiveInteger"/>

 <element name="SlotHandle" type="hexBinary"/>

 <element name="DIDName" type="cg:NameType"/>

 <element maxOccurs="1" minOccurs="0" name="PIN" type="string"/>

 <element name="Message" type="hexBinary"/>

 </sequence>

</complexType>

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

#### JSON Syntax

The SignRequestType JSON object SHALL implement in JSON syntax the requirements defined in the SignRequest component.

Properties of the JSON object SHALL implement the sub-components of SignRequest using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| MaxWaitSeconds | waitSecs | [] |
| SlotHandle | slot | [] |
| DIDName | did | [] |
| PIN | pin | [] |
| Message | msg | [] |

The SignRequestType JSON schema listed below for information.

"cg-SignRequestType": {

 "$xsd-full-type": "cg:SignRequestType dss2:SignRequestType",

 "type": "object",

 "properties": {

 "didname": {

 "type": "string"

 },

 "waitSecs": {

 "type": "integer"

 },

 "slot": {

 "type": "array",

 "items": {

 "type": "integer"

 }

 },

 "did": {

 "type": "string"

 },

 "pin": {

 "type": "string"

 },

 "msg": {

 "type": "array",

 "items": {

 "type": "integer"

 }

 }

 },

 "required": ["waitSecs", "slot", "did", "msg"]

}

[component SignRequest JSON schema details]

### Component SignResponse

#### Semantics

[component SignResponse normative details]

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

* The SessionIdentifier element MUST contain one instance of a string. [sub component SessionIdentifier details]
* The optional RetryCounter element MUST contain a non-negative integer. [sub component RetryCounter details]
* The optional Signature element MUST contain base64 encoded binary data. [sub component Signature details]

A set of sub-components is inherited from component Response and is not repeated here.

Non-normative Comment:

[component SignResponse non normative details]

#### XML Syntax

The XML type SignResponseType SHALL implement the requirements defined in the SignResponse component.

The SignResponseType 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.

<complexType name="SignResponseType">

 <complexContent>

 <extension base="cg:ResponseType">

 <sequence>

 <element name="SessionIdentifier" type="string"/>

 <element maxOccurs="1" minOccurs="0" name="RetryCounter" type="nonNegativeInteger"/>

 <element maxOccurs="1" minOccurs="0" name="Signature" type="base64Binary"/>

 </sequence>

 </extension>

 </complexContent>

</complexType>

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

#### JSON Syntax

The SignResponseType JSON object SHALL implement in JSON syntax the requirements defined in the SignResponse component.

Properties of the JSON object SHALL implement the sub-components of SignResponse using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| SessionIdentifier | sessId | [] |
| RetryCounter | retryCnt | [] |
| Signature | sig | [] |

The SignResponseType JSON schema listed below for information.

"cg-SignResponseType": {

 "$xsd-full-type": "cg:SignResponseType dss2:SignResponseType",

 "type": "object",

 "properties": {

 "result": {

 "$ref": "#/definitions/dsb-ResultType"

 },

 "sessId": {

 "type": "string"

 },

 "retryCnt": {

 "type": "integer"

 },

 "sig": {

 "type": "array",

 "items": {

 "type": "integer"

 }

 }

 },

 "required": ["sessId"]

}

[component SignResponse JSON schema details]

### Component Terminate

#### Semantics

[component Terminate normative details]

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

* The optional SessionIdentifier element MUST contain a string. [sub component SessionIdentifier details]

A set of sub-components is inherited from component Response and is not repeated here.

Non-normative Comment:

[component Terminate non normative details]

#### XML Syntax

The XML type TerminateType SHALL implement the requirements defined in the Terminate component.

The TerminateType 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.

<complexType name="TerminateType">

 <complexContent>

 <extension base="cg:ResponseType">

 <sequence>

 <element maxOccurs="1" minOccurs="0" name="SessionIdentifier" type="string"/>

 </sequence>

 </extension>

 </complexContent>

</complexType>

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

#### JSON Syntax

The TerminateType JSON object SHALL implement in JSON syntax the requirements defined in the Terminate component.

Properties of the JSON object SHALL implement the sub-components of Terminate using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| SessionIdentifier | sessId | [] |

The TerminateType JSON schema listed below for information.

"cg-TerminateType": {

 "$xsd-full-type": "cg:TerminateType",

 "type": "object",

 "properties": {

 "result": {

 "$ref": "#/definitions/dsb-ResultType"

 },

 "sessId": {

 "type": "string"

 }

 }

}

[component Terminate JSON schema details]

## Referenced Structure Models from DSS-X base

### Component Result

#### Semantics

[component Result normative details]

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

* The ResultMajor element MUST contain one instance of a URI. Its value is limited to an item of the following set:
urn:oasis:names:tc:dss:1.0:resultmajor:Success
urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError
urn:oasis:names:tc:dss:1.0:resultmajor:ResponderError
urn:oasis:names:tc:dss:1.0:resultmajor:InsufficientInformation
urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:resultmajor:Pending
[sub component ResultMajor details]
* The optional ResultMinor element MUST contain a URI. [sub component ResultMinor details]
* The optional ResultMessage element MUST contain sub-component. A given element MUST satisfy the requirements specified in section InternationalString. [sub component ResultMessage details]
* The optional ProblemReference element MUST contain a string. [sub component ProblemReference details]

Non-normative Comment:

[component Result non normative details]

#### XML Syntax

The XML element is defined in the XML namespace 'http://docs.oasis-open.org/dss/ns/base' .The XML type ResultType SHALL implement the requirements defined in the Result component.

The ResultType 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="ResultType">

 <xs:sequence>

 <xs:element name="ResultMajor">

 <xs:simpleType>

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

 <xs:enumeration value="urn:oasis:names:tc:dss:1.0:resultmajor:Success"/>

 <xs:enumeration value="urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError"/>

 <xs:enumeration value="urn:oasis:names:tc:dss:1.0:resultmajor:ResponderError"/>

 <xs:enumeration value="urn:oasis:names:tc:dss:1.0:resultmajor:InsufficientInformation"/>

 <xs:enumeration value="urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:resultmajor:Pending"/>

 </xs:restriction>

 </xs:simpleType>

 </xs:element>

 <xs:element minOccurs="0" name="ResultMinor" type="xs:anyURI"/>

 <xs:element minOccurs="0" name="ResultMessage" type="dsb:InternationalStringType"/>

 <xs:element minOccurs="0" name="ProblemReference" type="xs:string"/>

 </xs:sequence>

</xs:complexType>

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

#### JSON Syntax

The ResultType JSON object SHALL implement in JSON syntax the requirements defined in the Result component.

Properties of the JSON object SHALL implement the sub-components of Result using JSON-specific names mapped as shown in the table below.

|  |  |  |
| --- | --- | --- |
| Element | Implementing JSON member name | Comments |
| ResultMajor | maj | [] |
| ResultMinor | min | [] |
| ResultMessage | msg | [] |
| ProblemReference | pRef | [] |

The ResultType JSON schema listed below for information.

"dsb-ResultType": {

 "$xsd-full-type": "dsb:ResultType",

 "type": "object",

 "properties": {

 "maj": {

 "type": "string",

 "enum": ["urn:oasis:names:tc:dss:1.0:resultmajor:Success", "urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError", "urn:oasis:names:tc:dss:1.0:resultmajor:ResponderError", "urn:oasis:names:tc:dss:1.0:resultmajor:InsufficientInformation", "urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing:resultmajor:Pending"]

 },

 "min": {

 "type": "string"

 },

 "msg": {

 "$ref": "#/definitions/dsb-InternationalStringType"

 },

 "pRef": {

 "type": "string"

 }

 },

 "required": ["maj"]

}

[component Result JSON schema details]

### Component InternationalString

#### Semantics

[component InternationalString normative details]

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

* The value element MUST contain one instance of a string. [sub component value details]
* The lang element MUST contain one instance of a ISO language descriptor. [sub component lang details]

Non-normative Comment:

[component InternationalString non normative details]

#### XML Syntax

The XML element is defined in the XML namespace 'http://docs.oasis-open.org/dss/ns/base' .The XML type InternationalStringType SHALL implement the requirements defined in the InternationalString component.

The InternationalStringType 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="InternationalStringType">

 <xs:simpleContent>

 <xs:extension base="xs:string">

 <xs:attribute ref="xml:lang" use="required"/>

 </xs:extension>

 </xs:simpleContent>

</xs:complexType>

Each child element of InternationalStringType XML element SHALL implement in XML syntax the sub-component that has a name equal to its local name. The element 'value' is represented by the component's XML tag text content. [component InternationalString XML schema details]

#### JSON Syntax

The InternationalStringType JSON object SHALL implement in JSON syntax the requirements defined in the InternationalString component.

Properties of the JSON object SHALL implement the sub-components of InternationalString using JSON-specific names mapped as shown in the table below.

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

The InternationalStringType JSON schema listed below for information.

"dsb-InternationalStringType": {

 "$xsd-full-type": "dsb:InternationalStringType",

 "type": "object",

 "properties": {

 "value": {

 "type": "string"

 },

 "lang": {

 "type": "string"

 }

 },

 "required": ["lang"]

}

[component InternationalString JSON schema details]

## Element / JSON name lookup tables

The subsequent table allows to find the names of a component's element for a given JSON member name.

|  |  |
| --- | --- |
| JSON member name | mapped from element name |
| alg | DigestMethodAlgorithm |
| attRef | AttRef |
| attURI | AttRefURI |
| cardType | CardType |
| cert | Certificate |
| certFilter | CertificateFilter |
| certNeedsPin | NeedsPinForCertAccess |
| chall | Challenge |
| di | DigestInfo |
| did | DIDName |
| downloasdAddress | DownloadAddress |
| ID | ID |
| IDREF | IDREF |
| iss | Issuer |
| lang | Language |
| maj | ResultMajor |
| mimeType | MimeType |
| min | ResultMinor |
| minVer | MinimumVersion |
| msg | ResultMessageMessage |
| other | Other |
| pin | PIN |
| policy | ServicePolicyAppliedPolicyPolicy |
| pre | NamespacePrefix |
| pRef | ProblemReference |
| privNeedsPin | NeedsPinForPrivateKeyAccess |
| profile | ProfileAppliedProfile |
| protAuthPath | HasProtectedAuthPath |
| reqID | RequestID |
| respID | ResponseID |
| result | Result |
| retryCnt | RetryCounter |
| sessId | SessionIdentifier |
| sessionId | SessionIdentifier |
| sig | Signature |
| slot | SlotHandle |
| ssn | UniqueSSN |
| tokenInfo | TokenInfo |
| uri | NamespaceURI |
| usage | KeyUsage |
| value | DigestValueValue |
| ver | Version |
| waitSecs | MaxWaitSeconds |
| webOrigin | WebOrigin |

The subsequent table allows to find the abbreviated JSON member names for a given element name.

|  |  |
| --- | --- |
| Element | Implementing JSON member name |
| Algorithm | alg |
| AppliedPolicy | policy |
| AppliedProfile | profile |
| AttRef | attRef |
| AttRefURI | attURI |
| CardType | cardType |
| Certificate | cert |
| CertificateFilter | certFilter |
| Challenge | chall |
| DIDName | did |
| DigestInfo | di |
| DigestMethod | alg |
| DigestValue | value |
| DownloadAddress | downloasdAddress |
| HasProtectedAuthPath | protAuthPath |
| ID | ID |
| IDREF | IDREF |
| Issuer | iss |
| KeyUsage | usage |
| Language | lang |
| MaxWaitSeconds | waitSecs |
| Message | msg |
| MimeType | mimeType |
| MinimumVersion | minVer |
| NamespacePrefix | pre |
| NamespaceURI | uri |
| NeedsPinForCertAccess | certNeedsPin |
| NeedsPinForPrivateKeyAccess | privNeedsPin |
| Other | other |
| PIN | pin |
| Policy | policy |
| ProblemReference | pRef |
| Profile | profile |
| RequestID | reqID |
| ResponseID | respID |
| Result | result |
| ResultMajor | maj |
| ResultMessage | msg |
| ResultMinor | min |
| RetryCounter | retryCnt |
| ServicePolicy | policy |
| SessionIdentifier | sessIdsessionId |
| Signature | sig |
| SlotHandle | slot |
| TokenInfo | tokenInfo |
| UniqueSSN | ssn |
| Value | value |
| Version | ver |
| WebOrigin | webOrigin |

# Conformance

This profile defines two conformance levels:

* Level 1 ‑ “Basic”,
* Level 2 ‑ “Comprehensive”

## Level 1 – “Basic”

The conformance level “Basic” allows to return individual verification results for each signature contained in a VerifyRequest component. For this purpose the VerifyResponse component MUST contain in OptionalOutputs a VerificationReport element, as specified in Section 3.2. The VerificationReport element MUST contain an IndividualSignatureReport element (see Section 3.3) for each signature or time stamp (i.e. SignatureObject component) contained in the VerifyRequest component.

The Details component within IndividualSignatureReport MAY contain other elements, such as the Optional Outputs defined in Section 4.5 of **[DSSCore]**.

## Level 2 – “Comprehensive”

The conformance level “Comprehensive” comprises all requirements of conformance Level 1 (“Basic”), as explained in Section 4.1. Furthermore, the Details component within each IndividualReport MUST contain exactly one object-specific element, which documents the detailed verification results for the signatures or validation data under consideration. It is REQUIRED in this conformance level that certificate values and revocation values are included into the verification report if requested by the IncludeCertificateValues- and IncludeRevocationValues-element within the ReturnVerifcationReport component (cf. Section 3.1).

The object-specific detail elements defined in this specification are given as follows:

* DetailedSignatureReport (cf. Section 3.5) ‑ is used for the verification of (advanced) electronic signatures.
* IndividualTimeStampReport (cf. Section 3.5.5) – is used for the verification of individual time stamps according to **[RFC3161]**, which are not included in a signature.
* IndividualCertificateReport (cf. Section 3.5.6) – is used for the verification of individual certificates according to **[RFC5280]**, which are not included in a signature.
* IndividualAttributeCertificateReport (cf. Section 3.5.7) ‑ is used for the verification of individual attribute certificates according to **[RFC3281]**, which are not included in a signature.
* IndividualCRLReport (cf. Section 3.5.8) ‑ is used for the verification of individual CRLs according to **[RFC5280]**, which are not included in a signature.
* IndividualOCSPReport (cf. Section 3.5.9) ‑ is used for the verification of individual OCSP-responses according to **[RFC2560]**, which are not included in a signature.
* EvidenceRecordReport (cf. Section 3.5.10) – is used for the verification of evidence records according to **[RFC4998]**.

Other object-specific detail elements MAY be defined in other profiles.

1. Index

DateTime, 12

1. Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Editor** | **Changes Made** |
| [Rev number] | [Rev Date] | Andreas Kuehne and Stefan Hagen | Initial Draft version with feedback from the TC |