

2 **Advanced Electronic Signature**  
3 **Profiles of the OASIS Digital**  
4 **Signature Service Version 1.0**

5 **Committee Specification**

6 **13 February 2007**

7 **Specification URIs:**

8 **This Version:**

9 <http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-AdES-cs-v1.0-r1.html>

10 <http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-AdES-cs-v1.0-r1.pdf>

11 **Latest Version:**

12 <http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-AdES-cs-v1.0-r1.html>

13 <http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-AdES-cs-v1.0-r1.pdf>

14 **Technical Committee:**

15 OASIS Digital Signature Services TC

16 **Chair(s):**

17 Nick Pope, Thales eSecurity

18 Juan Carlos Cruellas, Centre d'aplicacions avançades d'Internet (UPC)

19 **Editor(s):**

20 Juan Carlos Cruellas, Centre d'aplicacions avançades d'Internet (UPC)

21 **Related work:**

22 This specification is related to:

- 23
  - oasis-dss-core-spec-cs-v1.0-r1

24 **Abstract:**

25 This document defines one abstract profile of the OASIS DSS protocols for the  
26 purpose of creating and verifying XML or CMS based Advanced Electronic  
27 Signatures. It also defines two concrete sub-profiles: one for creating and verifying  
28 XML Advanced Electronic Signatures and the other for creating and verifying CMS  
29 based Advanced Electronic Signatures.

30 **Status:**

31 This document was last revised or approved by the OASIS Digital Signature Services  
32 TC on the above date. The level of approval is also listed above. Check the current  
33 location noted above for possible later revisions of this document. This document is  
34 updated periodically on no particular schedule.

35 Technical Committee members should send comments on this specification to the  
36 Technical Committee's email list. Others should send comments to the Technical  
37 Committee by using the "Send A Comment" button on the Technical Committee's web  
38 page at <http://www.oasis-open.org/committees/dss>.  
39 For information on whether any patents have been disclosed that may be essential to  
40 implementing this specification, and any offers of patent licensing terms, please refer  
41 to the Intellectual Property Rights section of the Technical Committee web page  
42 (<http://www.oasis-open.org/committees/dss/ipr.php>).  
43 The non-normative errata page for this specification is located at [http://www.oasis-](http://www.oasis-open.org/committees/dss)  
44 [open.org/committees/dss](http://www.oasis-open.org/committees/dss).

---

## Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.

Copyright © OASIS® 1993–2007. All Rights Reserved. OASIS trademark, IPR and other policies apply.

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 paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document 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 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

The names "OASIS" are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

# Table of Contents

82	1	Introduction.....	7
83	1.1	Terminology.....	7
84	1.2	Normative References .....	7
85	1.3	Non-Normative References.....	8
86	1.4	Namespaces .....	8
87	2	Overview.....	9
88	3	Advanced Electronic Signature abstract profile .....	10
89	3.1	Overview .....	10
90	3.2	Profile Features.....	11
91	3.2.1	Scope .....	11
92	3.2.2	Relationship To Other Profiles.....	11
93	3.2.3	Signature Object .....	11
94	3.3	Profile of Signing Protocol.....	11
95	3.3.1	Element <SignRequest> .....	12
96	3.3.1.1	Element <OptionalInputs> .....	12
97	3.3.1.1.1	New Optional Inputs .....	12
98	3.3.1.1.1.1	Optional Input <SignatureForm>.....	12
99	3.3.1.1.2	Optional Inputs already defined in the Core .....	12
100	3.3.1.1.2.1	Optional Input <SignatureType> .....	12
101	3.3.1.1.2.2	Optional inputs <ClaimedIdentity> and <KeySelector> .....	13
102	3.3.1.1.2.3	Optional Input <SignedProperties> .....	13
103	3.3.1.1.2.3.1	Requesting SigningTime .....	13
104	3.3.1.1.2.3.2	Requesting CommitmentTypeIndication .....	14
105	3.3.1.1.2.3.3	Requesting SignatureProductionPlace .....	14
106	3.3.1.1.2.3.4	Requesting SignerRole .....	15
107	3.3.1.1.2.3.5	Requesting AllDataObjectsTimeStamp .....	15
108	3.3.1.1.2.3.6	Requesting DataObjectFormat.....	15
109	3.3.2	Element <SignResponse> .....	16
110	3.3.2.1	Element <SignatureObject> .....	16
111	3.3.2.2	Optional Outputs.....	16
112	3.4	Profile of Verifying Protocol.....	16
113	3.4.1	Element <VerifyRequest> .....	16
114	3.4.1.1	Attribute Profile .....	16
115	3.4.1.2	Element <SignatureObject> .....	16
116	3.4.1.3	Element <OptionalInputs> .....	16
117	3.4.1.3.1	Element <ReturnUpdatedSignature> .....	16
118	3.5	Element <VerifyResponse> .....	17
119	3.5.1.1	Element <OptionalOutputs> .....	17
120	3.5.1.1.1	Optional Output <UpdatedSignature> .....	17
121	4	XML Advanced Electronic Signatures concrete Profile.....	18
122	4.1	Overview .....	18
123	4.2	Profile features .....	18
124	4.2.1	Identifier .....	18
125	4.2.2	Scope .....	19
126	4.2.3	Relationship To Other Profiles.....	19
127	4.2.4	Signature Object .....	19
128	4.2.5	Transport Binding.....	19

129	4.2.6	Security Binding .....	19
130	4.3	Profile of Signing Protocol .....	19
131	4.3.1	Attribute Profile .....	19
132	4.3.2	Element <SignRequest> .....	20
133	4.3.2.1	Element <OptionalInputs> .....	20
134	4.3.2.1.1	New Optional Inputs .....	20
135	4.3.2.1.1.1	Element <SignatureForm> .....	20
136	4.3.2.1.2	Optional Inputs already defined in the Core .....	20
137	4.3.2.1.2.1	Optional Input <SignatureType> .....	20
138	4.3.2.1.2.2	Optional inputs < ClaimedIdentity> and <KeySelector> .....	20
139	4.3.2.1.2.3	Optional Input <SignedProperties> .....	20
140	4.3.2.1.2.3.1	Requesting SigningTime .....	20
141	4.3.2.1.2.3.2	Requesting CommitmentTypeIndication .....	20
142	4.3.2.1.2.3.3	Requesting SignatureProductionPlace .....	20
143	4.3.2.1.2.3.4	Requesting SignerRole .....	21
144	4.3.2.1.2.3.5	Requesting AllDataObjectTimeStamp .....	21
145	4.3.2.1.2.3.6	Requesting DataObjectFormat .....	21
146	4.3.2.1.2.3.7	Requesting <xades:IndividualDataObjectTimeStamp> .....	21
147	4.3.3	Element <SignResponse> .....	22
148	4.3.3.1	Element <SignatureObject> .....	22
149	4.4	Profile of Verifying Protocol .....	22
150	4.4.1	Element <VerifyRequest> .....	22
151	4.4.1.1	Attribute Profile .....	22
152	4.4.1.2	Element <SignatureObject> .....	22
153	4.4.1.3	Element <OptionalInputs> .....	23
154	4.4.1.3.1	Optional Output <ReturnUpdatedSignature> .....	23
155	4.4.2	Element <VerifyResponse> .....	23
156	4.4.2.1	Element <OptionalOutputs> .....	23
157	4.4.2.1.1	Optional Output <UpdatedSignature> .....	23
158	4.5	Profile Bindings .....	23
159	4.5.1	Transport Bindings .....	23
160	4.5.2	Security Bindings .....	23
161	4.5.2.1	Security Requirements .....	23
162	4.5.2.2	TLS X.509 Mutual Authentication .....	23
163	5	CMS-based Advanced Electronic Signature profile .....	24
164	5.1	Overview .....	24
165	5.2	Profile features .....	25
166	5.2.1	Identifier .....	25
167	5.2.2	Scope .....	25
168	5.2.3	Relationship To Other Profiles .....	25
169	5.2.4	Signature Object .....	25
170	5.2.5	Transport Binding .....	25
171	5.2.6	Security Binding .....	25
172	5.3	Profile of Signing Protocol .....	25
173	5.3.1	Element <SignRequest> .....	25
174	5.3.1.1	Attribute Profile .....	26
175	5.3.1.2	Element <OptionalInputs> .....	26
176	5.3.1.2.1	New Optional Inputs .....	26
177	5.3.1.2.1.1	Element <SignatureForm> .....	26
178	5.3.1.2.2	Optional Inputs already defined in the Core .....	26
179	5.3.1.2.2.1	Element <SignatureType> .....	26
180	5.3.1.2.2.2	Optional inputs < ClaimedIdentity> / <KeySelector> .....	26
181	5.3.1.2.2.3	Element <SignedProperties> .....	26

182	5.3.1.2.2.3.1	Requesting signing-time .....	26
183	5.3.1.2.2.3.2	Requesting commitment-type-indication .....	26
184	5.3.1.2.2.3.3	Requesting signer-location .....	26
185	5.3.1.2.2.3.4	Requesting signer-attributes .....	27
186	5.3.1.2.2.3.5	Requesting content-time-stamp .....	27
187	5.3.1.2.2.3.6	Requesting content-hints .....	27
188	5.3.2	Element <SignResponse> .....	27
189	5.3.2.1	Element <SignatureObject> .....	27
190	5.4	Profile of Verifying Protocol .....	27
191	5.4.1	Element <VerifyRequest> .....	27
192	5.4.1.1	Attribute Profile .....	27
193	5.4.1.2	Element <OptionalInputs> .....	28
194	5.4.1.2.1	Element <ReturnUpdatedSignature> .....	28
195	5.4.1.3	Element <SignatureObject> .....	28
196	5.4.2	Element <VerifyResponse> .....	28
197	5.4.2.1	Element <OptionalOutputs> .....	28
198	5.4.2.1.1	Element <UpdatedSignature> .....	28
199	5.5	Profile Bindings .....	28
200	5.5.1	Transport Bindings .....	28
201	5.5.2	Security Bindings .....	28
202	5.5.2.1	Security Requirements .....	28
203	5.5.2.2	TLS X.509 Mutual Authentication .....	28
204	6	XML timestamps in XAdES signatures .....	29
205	6.1	Generation and inclusion of XML timestamps .....	29
206	6.1.1	Profile for XAdES timestamp containers .....	29
207	6.1.2	XML timestamp within xades:IndividualDataObjectsTimeStamp .....	30
208	6.1.3	XML timestamp within xades:AllDataObjectsTimeStamp .....	30
209	6.1.4	XML timestamp within xades:SigAndRefsTimeStamp .....	30
210	6.1.5	XML timestamp within xades:RefsOnlyTimeStamp .....	31
211	6.1.6	XML timestamp within xades:ArchiveTimeStamp .....	31
212	6.2	Verification of XML timestamps .....	31
213	6.2.1	Verification of of xades:IndividualDataObjectsTimeStamp including a XML timestamp .....	32
214	6.2.2	Verification of xades:AllDataObjectsTimeStamp including a XML timestamp .....	32
215	6.2.3	Verification of xades:SigAndRefsTimeStamp including a XML timestamp .....	33
216	6.2.4	Verification of xades:RefsOnlyTimeStamp including a XML timestamp .....	34
217	6.2.5	Verification of xades:ArchiveTimeStamp including a XML timestamp .....	34
218			
219	7	Identifiers defined in this specification .....	36
220	7.1	Predefined advanced electronic signature forms identifiers .....	36
221	7.2	Result Identifiers .....	36
222	A.	Acknowledgements .....	38
223			

---

# 1 Introduction

The DSS signing and verifying protocols are defined in [DSSCore]. As defined in that document, the DSS protocols have a fair degree of flexibility and extensibility. This document defines an abstract profile for the use of the DSS protocols for creating and verifying XML and CMS-based Advanced Electronic Signatures as defined in [XAdES] and [CAdES]. This document also defines two concrete profiles derived from the abstract one: one for creating and verifying XAdES signatures and the other for creating and verifying CAdES signatures.

## 1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF RFC 2119 [RFC 2119]. These keywords are capitalized when used to unambiguously specify requirements over protocol features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

This specification uses the following typographical conventions in text: `<ns:Element>`, **Attribute**, **Datatype**, **OtherCode**.

## 1.2 Normative References

- [AdES-XSD] J. C. Cruellas et al. AdES Profile Schema, OASIS, February 2007.
- [CAdES] CMS Advanced Electronic Signatures. ETSI TS 101 733, January 2007.
- [Core-XSD] S. Drees et al. *DSS Schema*. OASIS, February 2007).
- [DSSCore] S. Drees et al. *Digital Signature Service Core Protocols and Elements*. OASIS, February 2007.
- [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- [RFC 2634] . Hoffman (ed.). Enhanced Security Services for S/MIME <http://www.ietf.org/rfc/rfc2634.txt>, , IETF RFC 2634 June 1999
- [RFC 3852] R. Housley. Cryptographic Message Syntax (CMS) , IETF RFC 3852, July 2004.
- [XAdES] Advanced Electronic Signatures. ETSI TS 101 733. March 2006.
- [XML-ns] T. Bray, D. Hollander, A. Layman. *Namespaces in XML*. <http://www.w3.org/TR/1999/REC-xml-names-19990114>, W3C Recommendation, January 1999.

265 [XMLSig] D. Eastlake et al. *XML-Signature Syntax and Processing*.  
266 <http://www.w3.org/TR/1999/REC-xml-names-19990114>, W3C Recommendation, February  
267 2002.

## 268 1.3 Non-Normative References

## 269 1.4 Namespaces

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

273 This schema is associated with the following XML namespace:

274 `urn:oasis:names:tc:dss:1.0:profiles:AdES:schema#`

275 Conventional XML namespace prefixes are used in this document:

- 276     o The prefix `dss:` (or no prefix) stands for the DSS core namespace [Core-XSD].
- 277     o The prefix `ds:` stands for the W3C XML Signature namespace [XMLSig].
- 278     o The prefix `xades:` stands for ETSI XML Advanced Electronic Signatures (XAdES)  
279         document [XAdES].

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



---

## 2 Overview

This document defines three profiles of the protocols specified in: “Digital Signature Services Core Protocol and Elements” [DSSCore].

The first one is an abstract profile defining messages for supporting the lifecycle of advanced electronic signatures. Both, XML and CMS-based advanced electronic signatures are supported by this profile.

One concrete profile, derived from the aforementioned abstract profile, gives support to the lifecycle of XML advanced electronic signatures as specified in [XAdES].

A second concrete profile, also derived from the abstract one, gives support to the lifecycle of CMS-based advanced electronic signatures as specified in [CAdES].

Implementations should implement one of the concrete profiles (or both) in order to request generation or validation of advanced electronic signatures in one of the two formats (or both).

---

## 3 Advanced Electronic Signature abstract profile

### 3.1 Overview

This abstract profile supports operations within each phase of the lifecycle of two types of advanced electronic signature:

- XML encoded signatures based on [XMLSig] such as specified in [XAdES].
- Binary encoded signatures based on [RFC 3852] such as specified in [CAAdES].

Henceforward, the document will use the term advanced signature when dealing with issues that affect to both types of signatures. The document will use XAdES or CAAdES signatures when dealing with issues that affect one or the other but not both of them.

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

- SignRequest. This operation supports requests for:
  - Generating predefined advanced signature forms as defined in [XAdES] and [CAAdES].
  - Generating XML signatures incorporating specific signed/unsigned properties whose combination does not fit any predefined XAdES signature form. In such cases, the form MUST have been defined in a proprietary specification and MUST be identified by one URI.
  - Generating CMS signatures incorporating specific signed/unsigned attributes whose combination does not fit any predefined [CAAdES] signature form. In such cases, the form MUST have been defined in a proprietary specification and MUST be identified by one URI.
- SignResponse. This operation supports delivery of:
  - Predefined advanced signature forms as defined in [XAdES] and [CAAdES].
  - XML signatures with specific properties whose combination does not fit any predefined XAdES signature form. In such cases, the form MUST have been defined in some other specification and MUST be identified by one URI.
  - CMS signatures incorporating specific signed attributes whose combination does not fit any predefined [CAAdES] signature form. In such cases, the form MUST have been defined in some other specification and MUST be identified by one URI.

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

- VerifyRequest. This operation supports requests for:
  - Verifying a predefined advanced signature form.
  - Verifying XML signatures incorporating specific properties whose combination does not fit any predefined XAdES signature form.
  - Verifying any of the signatures mentioned above PLUS updating them by addition of additional properties (time-stamps, validation data, etc) leading to a predefined XAdES form.

- 334           ○ Verifying CMS signatures incorporating specific attributes whose combination  
335           does not fit any predefined [CAAdES] signature form.
- 336           ○ Verifying any of the signatures mentioned above PLUS updating them by  
337           addition of additional attributes (time-stamps, validation data, etc) leading to a  
338           predefined [CAAdES] form.
- 339           ○ Verifying a long-term advanced signature in a certain point of time.
- 340       ○ VerifyResponse. This operation supports delivery of:
  - 341           ○ Advanced signature verification result of signatures mentioned above.
  - 342           ○ Advanced signature verification result PLUS the updated signatures as  
343           requested.
- 344   The material for each operation will clearly indicate the lifecycle phase it pertains to.

## 345   **3.2 Profile Features**

### 346   **3.2.1 Scope**

347   This document profiles the DSS signing and verifying protocols defined in [DSSCore].

### 348   **3.2.2 Relationship To Other Profiles**

349   The profile in this document is based on the [DSSCore]. The profile in this document may not  
350   be directly implemented. It is further profiled by the two concrete profiles also defined in  
351   sections 4 and 5.

### 352   **3.2.3 Signature Object**

353   This profile supports the creation and verification of advanced signatures as defined in  
354   [XAdES] and [CAAdES].

355   This profile also supports update of advanced signatures by addition of unsigned properties  
356   (time-stamps and different types of validation data), as specified in [XAdES] and [CAAdES].

## 357   **3.3 Profile of Signing Protocol**

358   The present profile allows requesting:

- 359       ○ Predefined forms of advanced electronic signatures as defined in [XAdES] and  
360       [CAAdES].
- 361       ○ Other forms of signatures based in [XMLSig] or [RFC 3852] defined in other  
362       specifications,

363   In both cases, the specific requested form will be identified by an URI.

364   According to this profile, the following predefined advanced signature forms defined in  
365   [XAdES] and [CAAdES] MAY be requested (those forms whose name begin by XAdES- are  
366   forms names for XAdES signatures; those ones whose name begin by CAAdES are names for  
367   CAAdES signatures):

- 368       ○ CAAdES-BES and XAdES-BES. In this form, the signing certificate is secured by the  
369       signature itself.

- 370 ○ CAdES-EPES and XAdES-EPES. This form incorporates an explicit identifier of the  
371 signature policy that will govern the signature generation and verification.
- 372 ○ CAdES-ES-T and XAdES-T. This form incorporates a trusted time, by means of a  
373 time-stamp token or a time-mark.
- 374 ○ CAdES-ES-C and XAdES-C.
- 375 ○ CAdES-ES-X and XAdES-X.
- 376 ○ CAdES-ES-X-L and XAdES-X-L.
- 377 ○ CAdES-ES-A and XAdES-A.

378 In addition, the present profile provides means for requesting incorporation in any of the  
379 aforementioned forms any of the signed properties defined in [XAdES] and signed attributes  
380 defined in [CAdES].

381 Other electronic signature forms based in [XMLSig] or [RFC 3852], defined elsewhere, MAY  
382 also be requested using the mechanisms defined in this profile.

### 383 **3.3.1 Element <SignRequest>**

384 This clause profiles the `dss:SignRequest` element.

#### 385 **3.3.1.1 Element <OptionalInputs>**

##### 386 **3.3.1.1.1 New Optional Inputs**

###### 387 **3.3.1.1.1.1 Optional Input <SignatureForm>**

388 The form of signature required MAY be indicated using the following new optional input

389 `<xs:element name="SignatureForm" type="xs:anyURI" />`

390 If not present the signature form SHALL be implied by the selected <SignaturePolicy> or  
391 the signature policy applied by the server.

392 Section 7.1 of this abstract profile defines a set of URIs identifying the predefined advanced  
393 electronic signature forms specified in [CAdES] and [XAdES].

394 Should other standard or proprietary specification define new signature forms and their  
395 corresponding URIs, concrete sub-profiles of this abstract profile could be defined for giving  
396 support to their verification and update.

397 Should a form identified by an URI, admit different properties combinations, the server will  
398 produce a specific combination depending on its policy or configuration settings.

##### 399 **3.3.1.1.2 Optional Inputs already defined in the Core**

400 None of the optional inputs specified in the [DSS Core] are precluded in this abstract profile. It  
401 only constrains some of them and specifies additional optional inputs.

###### 402 **3.3.1.1.2.1 Optional Input <SignatureType>**

403 This element is OPTIONAL. If present, <SignatureType> SHALL be either:

404 `urn:ietf:rfc:3275`

405 for requesting XML-based signatures, or

**urn:ietf:rfc:3369**

for requesting CMS-based signatures, as defined in 7.1 of [DSS Core].

If not present the signature type SHALL be implied by the selected <SignaturePolicy> or the signature policy applied by the server.

#### 3.3.1.1.2.2 Optional inputs <ClaimedIdentity> and <KeySelector>

As forms defined in [XAdES] and [CAAdES] require that the signing certificate is protected by the signature, the server MUST gain access to that certificate.

<dss:ClaimedIdentity> or <dss:KeySelector> optional inputs MAY be present. If they are not present, the server may use means not specified in this profile to identify the signer's key and gain access to its certificate.

#### 3.3.1.1.2.3 Optional Input <SignedProperties>

The requester MAY request to the server the addition of optional signed properties using the <dss:SignedProperties> element's <dss:Property> child profiled as indicated in clauses below. First names correspond to the one given by XAdES to the signed properties. Second ones correspond to the names given by CAAdES to the signed attributes.

Signed properties that MAY be requested are:

XAdES	CAAdES
SigningTime	signing-time
CommitmentTypeIndication	commitment-type-indication
SignerRole	signer-attributes
SignatureProductionPlace	signer-location
DataObjectFormat	content-hints
AllDataObjectsTimeStamp	content-time-stamp
IndividualDataObjectsTimeStamp	No equivalent signed attribute

Next sub-sections show how a client should request each of the aforementioned properties-attributes. The type of signature requested (XAdES or CAAdES) will determine whether a XAdES property or a CAAdES attribute is generated by the server.

#### 3.3.1.1.2.3.1 Requesting SigningTime

Value for <Identifier> element:

**urn:oasis:names:tc:dss:1.0:profiles:AdES:SigningTime**

If the client does not request such property, the server still MAY generate and include this property depending on its policy.

431 No content is required for `Value` element, since the actual contents of the property will be  
432 generated by the server when required.

#### 433 3.3.1.1.2.3.2 Requesting `CommitmentTypeIndication`

434 Value for `<Identifier>` element:

435 **`urn:oasis:names:tc:dss:1.0:profiles:AdES:CommitmentTypeIndication`**

436 If the client does not request such property, the server still MAY generate and include it with  
437 values that depend on server's policy.

438 The client MAY request the generation and inclusion of this signed property. In such cases  
439 the `<Value>` element MUST have the following content:

```
440 <xs:element name="RequestedCommitment">
441   <xs:complexType>
442     <xs:choice>
443       <xs:element ref="xades:CommitmentTypeIndication"/>
444       <xs:element name="BinaryValue" type="xs:base64Binary"/>
445     </xs:choice>
446   </xs:complexType>
447 </xs:element>
```

448 Element `<xades:CommitmentTypeIndication>` will be present when requesting a XML  
449 signature.

450 Element `<BinaryValue>` will be present when requesting an ASN.1 signature. Its contents  
451 MUST be the base64 encoding of `commitment-type-indication` ASN.1 attribute defined  
452 in [CADES], DER-encoded

#### 453 3.3.1.1.2.3.3 Requesting `SignatureProductionPlace`

454 Value for `<Identifier>` element:

455 **`urn:oasis:names:tc:dss:1.0:profiles:AdES:SignatureProductionPlace`**

456 The client MAY request a certain value for this property. Nevertheless, this value MAY be  
457 ignored by the server depending on its own policy, and the property be set to another value.

458 For requesting a value for this property, the `<Value>` element MUST have the following  
459 content:

```
460 <xs:element name="RequestedSignatureProductionPlace">
461   <xs:complexType>
462     <xs:choice>
463       <xs:element ref="xades:SignatureProductionPlace"/>
464       <xs:element name="BinaryValue" type="xs:base64Binary"/>
465     </xs:choice>
466   </xs:complexType>
467 </xs:element>
```

468 Element `<xades:SignatureProductionPlace>` will be present when requesting a XML  
469 signature.

470 Element `<BinaryValue>` will be present when requesting an ASN.1 signature. Its contents  
471 MUST be the base64 encoding of `signerLocation` ASN.1 attribute defined in [CADES],  
472 DER-encoded.

#### 3.3.1.1.2.3.4 Requesting SignerRole

Value for <Identifier> element:

**urn:oasis:names:tc:dss:1.0:profiles:AdES:SignerRole**

When the client requests the generation and inclusion of this signed property the <Value> element MUST have the following content:

```
<xs:element name="RequestedSignerRole">
  <xs:complexType>
    <xs:choice>
      <xs:element ref="xades:SignerRole"/>
      <xs:element name="BinaryValue" type="xs:base64Binary"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Element <xades:SignerRole> will be present when requesting a XML signature.

Element <BinaryValue> will be present when requesting a ASN.1 signature. Its contents MUST be the base64 encoding of **signer-attributes** ASN.1 attribute defined in [CADES], DER-encoded.

#### 3.3.1.1.2.3.5 Requesting AllDataObjectsTimeStamp

This element will be added for requesting the generation and inclusion of a time-stamp token on (all) the data object(s) to be signed.

Value for <Identifier> element:

**urn:oasis:names:tc:dss:1.0:profiles:AdES:AllDataObjectsTimeStamp**

No content is required for <Value> element, since the actual contents of the property will be generated by the server when required.

#### 3.3.1.1.2.3.6 Requesting DataObjectFormat

Value for Identifier element:

**urn:oasis:names:tc:dss:1.0:profiles:AdES:DataObjectFormat**

When the client requests the generation and inclusion of this signed property the <Value> element MUST have the following content.

```
<xs:element name="RequestedDocsFormat" type="DocsFormatType" />

<xs:complexType name="DocsFormatType">
  <xs:sequence>
    <xs:choice>
      <xs:element name="DocFormat" type="DocFormatType"
maxOccurs="unbounded" />
      <xs:element name="BinaryValue" type="xs:base64Binary"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="DocFormatType">
  <xs:complexContent>
    <xs:extension base="DocReferenceType">
      <xs:sequence>
```

```

519         <xs:element ref="xades:DataObjectFormat"/>
520     </xs:sequence>
521 </xs:extension>
522 </xs:complexContent>
523 </xs:complexType>

```

524 Elements <DocFormat> will be present when requesting an XML based signature.

525 Element <BinaryValue> will be present when requesting a CMS based signature. Its  
526 contents MUST be the base64 encoding of **content-hints** ASN.1 attribute defined in [RFC  
527 2634] DER-encoded.

### 528 3.3.2 Element <SignResponse>

529 This clause profiles the `dss:SignResponse` element.

#### 530 3.3.2.1 Element <SignatureObject>

531 This element **SHALL NOT** contain a `dss:TimeStamp` element as a child.

#### 532 3.3.2.2 Optional Outputs

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

## 535 3.4 Profile of Verifying Protocol

### 536 3.4.1 Element <VerifyRequest>

537 This clause specifies the profile for the contents of the `dss:VerifyRequest` when used for:

- 538 ○ Requesting verification of advanced signatures.
- 539 ○ Requesting verification of advanced signatures AND update of signatures to other  
540 predefined forms.

#### 541 3.4.1.1 Attribute Profile

542 The value for the `Profile` attribute, indicating the concrete sub-profile of this abstract profile,  
543 MUST be present.

#### 544 3.4.1.2 Element <SignatureObject>

545 This element **SHALL NOT** contain a `dss:TimeStamp` element as a child.

#### 546 3.4.1.3 Element <OptionalInputs>

547 None of the optional inputs specified in the [DSS Core] are precluded in this abstract profile. It  
548 only constrains some of them and specifies additional optional inputs.

#### 549 3.4.1.3.1 Element <ReturnUpdatedSignature>

550 This element **MUST** be present when the client requests verification of a signature and  
551 update to a predefined form of advanced signature.

552 The `Type` attribute identifies the advanced signature form requested.



553 Acceptable predefined values for this attribute are the URIs specified in table 1 corresponding  
554 to the following forms predefined in [CAAdES] and [XAdES]: XAdES-T/CAAdES-T, XAdES-  
555 C/CAAdES-C, XAdES-X/CAAdES-X, XAdES-X-L/CAAdES-X-L, XAdES-A/CAAdES-A.

556 Should other standard or proprietary specification define new signature forms and their  
557 corresponding URIs, concrete sub-profiles of this abstract profile could be defined for giving  
558 support to their verification and update.

559 When the requested form allows for different contents, the server MUST decide the specific  
560 contents of the updated signature delivered, according to its configuration and settings.

## 561 **3.5 Element <VerifyResponse>**

562 This clause profiles the `dss:VerifyResponse` element.

### 563 **3.5.1.1 Element <OptionalOutputs>**

564 None of the optional inputs specified in the [DSS Core] are precluded in this abstract profile. It  
565 only constrains some of them.

#### 566 **3.5.1.1.1 Optional Output <UpdatedSignature>**

567 This element SHALL contain a `dss:SignatureObject` element that SHALL NOT contain a  
568 `dss:TimeStamp` element as a child.

---

## 4 XML Advanced Electronic Signatures concrete Profile

### 4.1 Overview

This concrete profile supports operations within each phase of the lifecycle of XML Advanced Electronic Signature based on [XMLSig] such as specified in [XAdES]. It will then provide all the features related to XAdES signatures that are specified in the abstract profile defined in section 3.

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

- SignRequest. This operation supports requests for:
  - Generating predefined advanced signature forms as defined in [XAdES].
  - Generating XML signatures incorporating specific signed/unsigned properties whose combination does not fit any predefined XAdES signature form. In such cases, the form MUST have been defined in a proprietary specification and MUST be identified by one URI.
- SignResponse. This operation supports delivery of:
  - Predefined advanced signature forms as defined in [XAdES].
  - XML signatures with specific properties whose combination does not fit any predefined XAdES signature form. In such cases, the form MUST have been defined in a proprietary specification and MUST be identified by one URI.

For verification [and updating] of XAdES signatures the following operations apply:

- VerifyRequest. This operation supports requests for:
  - Verifying a predefined XAdES signature form.
  - Verifying XML signatures incorporating specific properties whose combination does not fit any predefined XAdES signature form.
  - Verifying any of the signatures mentioned above PLUS updating them by adding unsigned properties (time-stamps, validation data, etc) leading to a predefined XAdES form.
  - Verifying a long-term advanced signature in a certain point of time.
- VerifyResponse. This operation supports delivery of:
  - Advanced signature verification result of signatures mentioned above.
  - Advanced signature verification result PLUS the updated signatures as requested.

### 4.2 Profile features

#### 4.2.1 Identifier

urn:oasis:names:tc:dss:1.0:profiles:XAdES.

## 4.2.2 Scope

This document profiles the DSS abstract profile defined in section 3 of the present document.

## 4.2.3 Relationship To Other Profiles

The profile in this section is based on the abstract profile for Advanced Electronic Signatures defined in section 3.

## 4.2.4 Signature Object

This profile supports the creation and verification of XML advanced signatures as defined in [XAdES].

This profile also supports verification and update of advanced signatures by addition of unsigned properties (time-stamps and different types of validation data), as specified in [XAdES]

## 4.2.5 Transport Binding

This profile does not specify or constrain the transport binding.

## 4.2.6 Security Binding

This profile does not specify or constrain the security binding.

## 4.3 Profile of Signing Protocol

The present profile allows requesting:

- Predefined forms of advanced electronic signatures as defined in [XAdES]. A server aligned with this profile SHALL generate XAdES signatures with direct incorporation of qualifying properties as defined in [XAdES] section 6.3.
- Other forms of signatures based in [XMLSig] defined in other specifications,

In both cases, the specific requested form will be identified by an URI.

According to this profile, the following predefined advanced signature forms defined in [XAdES] MAY be requested: XAdES-BES, XAdES-EPES, XAdES-T, XAdES-C, XAdES-X, XAdES-X-L., and XAdES-A.

In addition, the present profile provides means for requesting incorporation in any of the aforementioned forms any of the following properties: *SigningTime*, *CommitmentTypeIndication*, *SignatureProductionPlace*, *SignerRole*, *IndividualDataObjectTimeStamp*, *AllDataObjectTimeStamp* and *DataObjectFormat*.

Other electronic signature forms based in [XMLSig] defined elsewhere MAY also be requested using the mechanisms defined in this profile.

### 4.3.1 Attribute Profile

**urn:oasis:names:tc:dss:1.0:profiles:XAdES.**

## 4.3.2 Element <SignRequest>

This clause profiles the `dss:SignRequest` element.

### 4.3.2.1 Element <OptionalInputs>

#### 4.3.2.1.1 New Optional Inputs

##### 4.3.2.1.1.1 Element <SignatureForm>

Usage of these elements is according to what is stated in section 3.3.1.1.1.1.

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

None of the optional inputs specified in the [DSS Core] are precluded in this abstract profile. It only constrains some of them and specifies additional optional inputs.

##### 4.3.2.1.2.1 Optional Input <SignatureType>

This element is MANDATORY. Its value MUST be:

```
urn:ietf:rfc:3275
```

##### 4.3.2.1.2.2 Optional inputs <ClaimedIdentity> and <KeySelector>

Usage of these elements is according to what is stated in section 3.3.1.1.2.2.

##### 4.3.2.1.2.3 Optional Input <SignedProperties>

###### 4.3.2.1.2.3.1 Requesting SigningTime

Clients MAY use the URI defined in 3.3.1.1.2.3.1 or alternatively they MAY also use the following one:

```
urn:oasis:names:tc:dss:1.0:profiles:XAdES:SigningTime
```

Usage of these elements is according to what is stated in section 3.3.1.1.2.3.1.

###### 4.3.2.1.2.3.2 Requesting CommitmentTypeIndication

Clients MAY use the URI defined in 3.3.1.1.2.3.2 or alternatively they MAY also use the following one:

```
urn:oasis:names:tc:dss:1.0:profiles:XAdES:CommitmentTypeIndication
```

When this optional input is present, the <Value> element MUST contain a <RequestedCommitment> element as defined in section 3.3.1.1.2.3.2 with the <xades:CommitmentTypeIndication>.

###### 4.3.2.1.2.3.3 Requesting SignatureProductionPlace

Clients MAY use the URI defined in 3.3.1.1.2.3.3 or alternatively they MAY also use the following one:

```
urn:oasis:names:tc:dss:1.0:profiles:XAdES:SignatureProductionPlace
```

When this optional input is present, the <Value> element MUST contain a <RequestedSignatureProductionPlace> element as defined in section 3.3.1.1.2.3.3 with the <xades:SignatureProductionPlace>.

#### 4.3.2.1.2.3.4 Requesting SignerRole

Clients MAY use the URI defined in 3.3.1.1.2.3.4 or alternatively they MAY also use the following one:

**urn:oasis:names:tc:dss:1.0:profiles:XAdES:SignerRole**

When this optional input is present, the <Value> element MUST contain a <RequestedSignerRole> element as defined in section 3.3.1.1.2.3.4 with the <xades:SignerRole> child.

#### 4.3.2.1.2.3.5 Requesting AllDataObjectTimeStamp

Clients MAY use the URI defined in 3.3.1.1.2.3.5 or alternatively they MAY also use the following one:

**urn:oasis:names:tc:dss:1.0:profiles:XAdES:AllDataObjectsTimeStamp**

Usage of these elements is according to what is stated in section 3.3.1.1.2.3.5.

#### 4.3.2.1.2.3.6 Requesting DataObjectFormat

Clients MAY use the URI defined in 3.3.1.1.2.3.6 or alternatively they MAY also use the following one:

**urn:oasis:names:tc:dss:1.0:profiles:XAdES:AllDataObjectsTimeStamp**

When this optional input is present, the <Value> element MUST contain a <RequestedDocsFormat> element as defined in section 3.3.1.1.2.3.6 with one or more <DocFormat> children.

#### 4.3.2.1.2.3.7 Requesting <xades:IndividualDataObjectTimeStamp>

Value for <Identifier> element:

**urn:oasis:names:tc:dss:1.0:profiles:XAdES:IndividualDataObjectTimeStamp**

In this case, the content of <Value> element will be the element <DocsToBeTimeStamped>, defined as shown below.

```
<xs:element name="DocsToBeTimeStamped" type="DocReferencesType" />
<xs:complexType name="DocReferencesType">
  <xs:sequence>
    <xs:element name="DocReference" maxOccurs="unbounded"
      type="DocReferenceType" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="DocReferenceType">
  <xs:attribute name="WhichDocument" type="xs:IDREF"
    use="required" />
  <xs:attribute name="RefId" type="xs:string" use="optional" />
</xs:complexType>
```

WhichDocument attribute contains the reference to the document whose time-stamp is requested (see attribute ID in [CoreDSS] section 2.4.1). Should the client request the generation of several ds:Reference element for this document (using dss:SignedReferences optional input), the server SHALL timestamp all the data objects referenced by these ds:Reference elements. Under these conditions, each dss:SignedReference element MUST have its RefId attribute set to a not empty value.

[XAdES] mandates that <ds:Reference> elements corresponding to signed data objects that have been individually time-stamped before being signed, must include an Id attribute. [XAdES] also mandates <xades:IndividualDataObjectsTimeStamp> element to use this Id attribute to indicate what signed documents have actually been time-stamped before signing. See [XAdES] <xades:TimeStampType> and <xades:IndividualDataObjectsTimeStamp> definitions for more details.

The client MAY request a value for the <ds:Reference> element's Id attribute using the RefId optional attribute if a <dss:SignedReference> forcing a value for such an attribute is not present in the request. If the request does not specify a value for this attribute, then the server will automatically generate it.

### 4.3.3 Element <SignResponse>

This section profiles the dss:SignResponse element.

#### 4.3.3.1 Element <SignatureObject>

The content of this element MUST be one of the following:

A ds:Signature element containing a XMLSig based signature.

A dss:SignaturePtr pointing to the XMLSig based signature embedded in an output document.

## 4.4 Profile of Verifying Protocol

A server verifying XAdES signatures SHOULD follow the recommendations made by the XAdES standard it aligns to with respect on how to verify the signed and unsigned properties (version XAdES v1.3.2 includes an informative annex on this topic).

### 4.4.1 Element <VerifyRequest>

This clause profiles the dss:VerifyRequest element.

#### 4.4.1.1 Attribute Profile

urn:oasis:names:tc:dss:1.0:profiles:XAdES.

#### 4.4.1.2 Element <SignatureObject>

This element SHALL NOT contain a dss:TimeStamp element as a child.

#### 4.4.1.3 Element <OptionalInputs>

##### 4.4.1.3.1 Optional Output <ReturnUpdatedSignature>

Usage of these elements is according to what is stated in section 3.4.1.3.1.

#### 4.4.2 Element <VerifyResponse>

This clause profiles the `dss:VerifyResponse` element.

##### 4.4.2.1 Element <OptionalOutputs>

None of the optional inputs specified in the [DSS Core] are precluded in this profile. It only constrains some of them.

##### 4.4.2.1.1 Optional Output <UpdatedSignature>

The content of the `dss:UpdatedSignature` will be a `dss:SignatureObject` element with one of the following contents:

- A `ds:Signature` containing a XMLSig based signature.
- A `dss:SignaturePtr` pointing to the XMLSig based signature embedded in one of the inputdocuments.

### 4.5 Profile Bindings

#### 4.5.1 Transport Bindings

Messages transported in this profile MAY be transported by the HTTP POST Transport Binding and the SOAP 1.2 Transport Binding defined in [DSSCore].

#### 4.5.2 Security Bindings

##### 4.5.2.1 Security Requirements

This profile MUST use security bindings that:

- Authenticates the requester to the DSS server
- Authenticates the DSS server to the DSS client
- Protects the integrity of a request, response and the association of response to the request.
- Optionally, protects the confidentiality of a request and response.
- The following MAY be used to meet these requirements.

##### 4.5.2.2 TLS X.509 Mutual Authentication

This profile is secured using the TLS X.509 Mutual Authentication Binding defined in [DSSCore].

---

## 5 CMS-based Advanced Electronic Signature profile

### 5.1 Overview

This concrete profile supports operations within each phase of the lifecycle of CMS based Advanced Electronic Signature based on [RFC 3852] such as specified in [CAAdES]. It will then provide all the features related to CAAdES signatures that are specified in the abstract profile defined in section 3.

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

- SignRequest. This operation supports requests for:
  - Generating predefined advanced signature forms as defined in [CAAdES].
  - Generating CMS signatures incorporating specific signed/unsigned attributes whose combination does not fit any predefined [CAAdES] signature forms. In such cases, the form MUST have been defined in a proprietary specification and MUST be identified by one URI.
- SignResponse. This operation supports delivery of:
  - Predefined advanced signature forms as defined in [CAAdES].
  - CMS signatures incorporating specific signed attributes whose combination does not fit any predefined [CAAdES] signature forms. In such cases, the form MUST have been defined in a proprietary specification and MUST be identified by one URI.

For verification [and updating] of signatures as specified in [CAAdES] the following operations apply:

- VerifyRequest. This operation supports requests for:
  - Verifying a predefined [CAAdES] signature form.
  - Verifying CMS signatures incorporating specific attributes whose combination does not fit any predefined [CAAdES] signature form.
  - Verifying any of the signatures mentioned above PLUS updating them by addition of additional attributes (time-stamps, validation data, etc) leading to a predefined [CAAdES] form.
  - Verifying a long-term advanced signature in a certain point of time.
- VerifyResponse. This operation supports delivery of:
  - Advanced signature verification result of signatures mentioned above.
  - Advanced signature verification result PLUS the updated signatures as requested.



## 5.2 Profile features

### 5.2.1 Identifier

urn:oasis:names:tc:dss:1.0:profiles:CAAdES.

### 5.2.2 Scope

This document profiles the DSS abstract profile defined in section 3 of the present document.

### 5.2.3 Relationship To Other Profiles

The profile in this document is based on the abstract profile for Advanced Electronic Signatures defined in section 3.

### 5.2.4 Signature Object

This profile supports the creation and verification of CMS based advanced signatures as defined in [CAAdES].

This profile also supports verification and update of advanced signatures by addition of unsigned properties (time-stamps and different types of validation data), as specified in [CAAdES]

### 5.2.5 Transport Binding

This profile does not specify or constrain the transport binding.

### 5.2.6 Security Binding

This profile does not specify or constrain the security binding.

## 5.3 Profile of Signing Protocol

The present profile allows requesting:

- Predefined forms of advanced electronic signatures as defined in [CAAdES].
- Other forms of signatures based in [RFC 3852] defined in other specifications,

In both cases, the specific requested form will be identified by an URI.

According to this profile, the following predefined advanced signature forms defined in [CAAdES] MAY be requested: CAAdES-BES, CAAdES-EPES, CAAdES-T, CAAdES-C, CAAdES-X, CAAdES-X-L, and CAAdES-A

In addition, the present profile provides means for requesting incorporation in any of the aforementioned forms any of the following attributes: **signing-time**, **commitment-type-indication**, **signer-attributes**, **signer-location**, **content-hints**, and **content-time-stamp**

Other electronic signature forms based in [RFC 3852], defined elsewhere, MAY also be requested using the mechanisms defined in this profile.

### 5.3.1 Element <SignRequest>

This clause profiles the `dss:SignRequest` element.

### 5.3.1.1 Attribute Profile

`urn:oasis:names:tc:dss:1.0:profiles:CAAdES.`

### 5.3.1.2 Element <OptionalInputs>

#### 5.3.1.2.1 New Optional Inputs

##### 5.3.1.2.1.1 Element <SignatureForm>

Usage of these elements is according to what is stated in 3.3.1.1.1.1.

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

None of the optional inputs specified in the [DSS Core] are precluded in this abstract profile. It only constrains some of them and specifies additional optional inputs.

##### 5.3.1.2.2.1 Element <SignatureType>

This element is MANDATORY. Its value MUST be:

`urn:ietf:rfc:3369`

##### 5.3.1.2.2.2 Optional inputs <ClaimedIdentity> / <KeySelector>

Usage of these elements is according to what is stated in section 3.3.1.1.2.2.

##### 5.3.1.2.2.3 Element <SignedProperties>

This section profiles section 3.3.1.1.2.3.

##### 5.3.1.2.2.3.1 Requesting signing-time

Clients MAY use the URI defined in 3.3.1.1.2.3.1 or alternatively they MAY also use the following one:

`urn:oasis:names:tc:dss:1.0:profiles:CAAdES:signing-time`

Usage of these elements is according to what is stated in section 3.3.1.1.2.3.1.

##### 5.3.1.2.2.3.2 Requesting commitment-type-indication

Clients MAY use the URI defined in 3.3.1.1.2.3.2 or alternatively they MAY also use the following one:

`urn:oasis:names:tc:dss:1.0:profiles:CAAdES:commitment-type-indication`

When this optional input is present, the <Value> element MUST contain a <RequestedCommitment> element as defined in section 3.3.1.1.2.3.2 with the <BinaryValue> child containing the base64encoding of `commitment-type-indication` ASN.1 attribute as specified in [CAAdES], DER-encoded.

##### 5.3.1.2.2.3.3 Requesting signer-location

Clients MAY use the URI defined in 3.3.1.1.2.3.3 or alternatively they MAY also use the following one:

`urn:oasis:names:tc:dss:1.0:profiles:CAAdES:signer-location`

When this optional input is present, the <Value> element MUST contain a <RequestedSignatureProductionPlace> element as defined in section 3.3.1.1.2.3.3 with the <BinaryValue> child containing the base64 encoding of **signer-location** ASN.1 attribute as specified in [CAAdES], DER-encoded.

#### 5.3.1.2.2.3.4 Requesting signer-attributes

Clients MAY use the URI defined in 3.3.1.1.2.3.4 or alternatively they MAY also use the following one:

**urn:oasis:names:tc:dss:1.0:profiles:CAAdES:signer-attributes**

When this optional input is present, the <Value> element MUST contain a <RequestedSignerRole> element as defined in section 3.3.1.1.2.3.4 with the <BinaryValue> child containing the base64 encoding of **signer-attributes** ASN.1 attribute as specified in [CAAdES], DER-encoded.

#### 5.3.1.2.2.3.5 Requesting content-time-stamp

Clients MAY use the URI defined in 3.3.1.1.2.3.5 or alternatively they MAY also use the following one:

**urn:oasis:names:tc:dss:1.0:profiles:CAAdES:content-time-stamp**

Usage of these elements is according to what is stated in section 3.3.1.1.2.3.5

#### 5.3.1.2.2.3.6 Requesting content-hints

Clients MAY use the URI defined in 3.3.1.1.2.3.6 or alternatively they MAY also use the following one:

**urn:oasis:names:tc:dss:1.0:profiles:CAAdES:content-hints**

When this optional input is present, the <Value> element MUST contain a <RequestedDocsFormat> element as defined in section 3.3.1.1.2.3.6 with the <BinaryValue> child containing the base64 encoding of **content-hints** ASN.1 attribute as specified in [CAAdES], DER-encoded.

### 5.3.2 Element <SignResponse>

This section profiles the `dss:SignResponse` element.

#### 5.3.2.1 Element <SignatureObject>

The `dss:SignatureObject` MUST contain the `dss:Base64Signature` child with a CMS based signature base-64 encoded.

## 5.4 Profile of Verifying Protocol

### 5.4.1 Element <VerifyRequest>

This clause profiles the `dss:VerifyRequest` element.

#### 5.4.1.1 Attribute Profile

`urn:oasis:names:tc:dss:1.0:profiles:CAAdES.`

### 5.4.1.2 Element <OptionalInputs>

#### 5.4.1.2.1 Element <ReturnUpdatedSignature>

Usage of these elements is according to what is stated in section 3.4.1.3.1.

### 5.4.1.3 Element <SignatureObject>

The `dss:SignatureObject` element MUST contain the `dss:Base64Signature` child with a CMS based signature base64 encoded.

## 5.4.2 Element <VerifyResponse>

This clause profiles the `dss:VerifyResponse` element.

### 5.4.2.1 Element <OptionalOutputs>

Usage of these elements is according to what is stated in section 3.5.1.1.

#### 5.4.2.1.1 Element <UpdatedSignature>

- The content of the `dss:UpdatedSignature` will be a `dss:SignatureObject` element with a `dss:Base64Signature` element with the CMS based signature base64 encoded.

## 5.5 Profile Bindings

### 5.5.1 Transport Bindings

Messages transported in this profile MAY be transported by the HTTP POST Transport Binding and the SOAP 1.2 Transport Binding defined in [DSSCore].

### 5.5.2 Security Bindings

#### 5.5.2.1 Security Requirements

This profile MUST use security bindings that:

- Authenticates the requester to the DSS server
- Authenticates the DSS server to the DSS client
- Protects the integrity of a request, response and the association of response to the request.
- Optionally, protects the confidentiality of a request and response.
- The following MAY be used to meet these requirements.

#### 5.5.2.2 TLS X.509 Mutual Authentication

This profile is secured using the TLS X.509 Mutual Authentication Binding defined in [DSSCore].

---

## 6 XML timestamps in XAdES signatures

XAdES specification [XAdES] defines a placeholder for incorporating XML timestamps within XAdES signatures. As at the time [XAdES] was written no XML timestamps had been specified, no details on their structure and management were included.

The current section provides rules for including XML timestamps into XAdES signatures. For the rest of the present document a XML timestamp is a `dss:Timestamp` element as defined in [DSSCore] section 5.1, incorporating a `ds:Signature` element profiled as indicated in [DSSCore] section 5.1.1.

### 6.1 Generation and inclusion of XML timestamps

#### 6.1.1 Profile for XAdES timestamp containers

[XAdES] defines the following timestamps containers:

`xades:IndividualDataObjectTimeStamp`, `xades:AllDataObjectTimeStamp`,  
`xades:SignatureTimeStamp`, `xades:RefsOnlyTimeStamp`,  
`xades:SigAndRefsTimeStamp` and `xades:ArchiveTimeStamp`.

XAdES timestamp containers MAY include more than one XML timestamp.

XAdES timestamp containers including XML timestamps will not use the explicit referencing mechanism (the `xades:Include` element) defined in [XAdES] section 7.1.4.3.1.

The current document defines the structure of XML timestamps that timestamp more than one item in XAdES signatures i.e., all the timestamps defined in XAdES except the signature timestamp, which has already been profiled in [DSSCore] section 3.5.2.2.

### 6.1.2 XML timestamp within `xades:IndividualDataObjectsTimeStamp`

This timestamp will be included within `xades:IndividualDataObjectsTimeStamp`'s `xades:XMLTimeStamp` child.

This timestamp must be compliant with the profile defined in [DSSCore] section 5.1.1.

In addition, this timestamp MUST include within its `ds:SignedInfo` one or more `ds:Reference` elements that will be built as indicate below.

1. Take all the XAdES signature's `ds:Reference` referencing those data objects designated by `dss:DocsToBeTimestamped`.
2. For each one proceed as indicated below:
  - a. Generate a copy.
  - b. Suppress the `Id` attribute of the copy if present.
  - c. Set the `type` attribute of the copy to the following URI:  
`http://uri.etsi.org/01903/#IndividualDataObjectsTimeStamp`.
  - d. Add the copy to the timestamp's `ds:SignedInfo`.

Applications compliant with the present profile MUST dereference all the `ds:Reference` elements within XML timestamp's `ds:SignedInfo` as indicated in [XMLSig]

### 6.1.3 XML timestamp within `xades:AllDataObjectsTimeStamp`

This timestamp will be included within `xades:AllDataObjectsTimeStamp`'s `xades:XMLTimeStamp` child.

This timestamp must be compliant with the profile defined in [DSSCore] section 5.1.1.

In addition, this timestamp MUST include one `ds:Reference` element without `URI` attribute and the `type` attribute set to the following URI.

`http://uri.etsi.org/01903/#AllDataObjectsTimeStamp`

It MUST NOT have any `ds:Transforms` element.

Applications compliant with the present profile MUST dereference this element by processing, as indicated in [XAdES] section 7.2.9 steps 1 to 3, all the `ds:Reference` elements in XAdES' `ds:SignedInfo`, except the one referencing the `xades:SignedProperties` element.

### 6.1.4 XML timestamp within `xades:SigAndRefsTimeStamp`

This timestamp will be included within `xades:SigAndRefsTimeStamp`'s `xades:XMLTimeStamp` child.

This timestamp must be compliant with the profile defined in [DSSCore] section 5.1.1.

In addition, this timestamp MUST include one `ds:Reference` element without `URI` attribute and the `type` attribute set to the following URI.

`http://uri.etsi.org/01903/#SigAndRefsTimeStamp`

1000 It MUST NOT have any `ds:Transforms` element.  
1001 Applications compliant with the present profile MUST dereference this element by taking the  
1002 data objects listed in [XAdES] section 7.5.1.1 and process them as indicated there.

### 1003 **6.1.5 XML timestamp within `xades:RefsOnlyTimeStamp`**

1004 This timestamp will be included within `xades:RefsOnlyTimeStamp`'s  
1005 `xades:XMLTimeStamp` child.  
1006 This timestamp must be compliant with the profile defined in [DSSCore] section 5.1.1.  
1007 In addition, this timestamp MUST include one `ds:Reference` element without `URI` attribute  
1008 and the `type` attribute set to the following URI.  
1009 <http://uri.etsi.org/01903/#RefsOnlyTimeStamp>  
1010 It MUST NOT have any `ds:Transforms` element.  
1011 Applications compliant with the present profile MUST dereference this element by the data  
1012 objects listed in [XAdES] section 7.5.2.1 and process them as indicated there.

### 1013 **6.1.6 XML timestamp within `xades:ArchiveTimeStamp`**

1014 This timestamp will be included within `xades:ArchiveTimeStamp`'s  
1015 `xades:XMLTimeStamp` child.  
1016 This timestamp must be compliant with the profile defined in [DSSCore] section 5.1.1.  
1017 In addition, this timestamp MUST include one `ds:Reference` element without `URI` attribute  
1018 and the `type` attribute set to the following URI.  
1019 <http://uri.etsi.org/01903/#ArchiveTimeStamp>  
1020 It MUST NOT have any `ds:Transforms` element.  
1021 Applications compliant with the present profile MUST dereference this element by taking the  
1022 data objects listed in [XAdES] section 7.7.1 and process them as indicated there.

## 1023 **6.2 Verification of XML timestamps**

1024 This section specifies the steps to be performed by a server for verifying the XML timestamps  
1025 present in a XAdES signature.  
1026 The steps that the server shall perform for initiating the verification of each XML timestamp  
1027 within the corresponding container are listed in order below (if any one of them results in  
1028 failure, then the timestamp token SHOULD be rejected).  
1029 1. Extract the timestamp token embedded in the incoming signature.  
1030 2. Verify that the verification key and algorithms used conforms to all relevant aspects of the  
1031 applicable policy. Should this key come within a public certificate, verify that the certificate  
1032 conforms to all relevant aspects of the applicable policy including algorithm usage, policy  
1033 OIDs, and time accuracy tolerances.  
1034 3. Verify that the aforementioned verification key is consistent with the  
1035 `ds:SignedInfo/SignatureMethod/@Algorithm` attribute value.  
1036 4. Verify the timestamp token signature in accordance with the rules defined in [XMLDSIG].  
1037 5. Verify that the `ds:SignedInfo` element contains only two `ds:Reference` elements



1038 6. Verify that one of the `ds:Reference` elements has its `Type` attribute set to  
1039 "urn:oasis:names:tc:dss:1.0:core:schema:XMLTimeStampToken". Take this one and  
1040 proceed as indicated below:

- 1041 a. Retrieve the referenced data object. Verify that it references a `ds:Object`  
1042 element, which in turn envelopes a `dss:TSTInfo` element.
- 1043 b. Verify that the `dss:TSTInfo` element has a valid layout as per the present  
1044 specification.
- 1045 c. Extract the digest value and associated algorithm from its `<ds:DigestValue>`  
1046 and `<ds:DigestMethod>` elements respectively.
- 1047 d. Recalculate the digest of the retrieved data object as specified by [XMLDSIG]  
1048 with the digest algorithm indicated in `<ds:DigestMethod>`, and compare this  
1049 result with the contents of `<ds:DigestValue>`.

1050 Subsequent sub-sections indicate the steps that the server shall perform for completing the  
1051 verification of each XML timestamp.

### 1052 **6.2.1 Verification of of xades:IndividualDataObjectsTimeStamp** 1053 **including a XML timestamp**

1054 After completing steps 1 to 5 in section 6.2., the server will perform the tasks detailed below  
1055 for completing the XML timestamp verification. If any one of them results in failure, then the  
1056 timestamp token SHOULD be rejected. For each of the remaining `ds:Reference` proceed  
1057 as indicated below:

- 1058 1. Check that it has been built from one of the `ds:Reference` elements within XAdES  
1059 signature applying the changes mentioned in section 6.1.2
- 1060 2. Dereference and validate it according to the rules stated in [XMLSig].
- 1061 3. Check for coherence in the value of the times indicated in the time-stamp tokens. All the  
1062 time instants must be previous to the time when the verification is performed, to the time  
1063 indicated within the `SigningTime` if present, and to the times indicated within the  
1064 time-stamp tokens enclosed within all the rest of time-stamp container properties except  
1065 other `IndividualDataObjectsTimeStamp`.
- 1066 4. Set the `<dss:Result>` element as appropriate.

1067 Minor Error

1068 `urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:InvalidIndivid`  
1069 `ualDataObjectsTimeStamp` MUST be used when the cryptographic signature verification  
1070 succeeds but this timestamp verification fails.

### 1071 **6.2.2 Verification of xades:AllDataObjectsTimeStamp including a** 1072 **XML timestamp**

1073 After completing steps 1 to 5 in section 6.2., the server will perform the steps listed below for  
1074 completing the XML timestamp verification. If any one of them results in failure, then the  
1075 timestamp token SHOULD be rejected.

- 1076 1. Take the other `ds:Reference` element and proceed to dereference it as indicated  
1077 below:
  - 1078 a. Take the first `ds:Reference` element within the XAdES signature's  
1079 `ds:SignedInfo` element if and only if the `Type` attribute doesn't have the value  
1080 "<http://uri.etsi.org/01903#SignedProperties>".



- 1081           b. Process it according to the reference processing model of XMLDSIG.  
 1082           c. If the result is a node-set, canonicalize it using the algorithm indicated in  
 1083           CanonicalizationMethod element of the property, if present. If not, the standard  
 1084           canonicalization method as specified by XMLDSIG must be used.  
 1085           d. Concatenate the resulting bytes in an octet stream.  
 1086           e. Repeat steps a) to d) for all the subsequent ds:Reference elements (in their order  
 1087           of appearance) within the XAdES signature's ds:SignedInfo element if and  
 1088           only if Type attribute has not the value  
 1089           "http://uri.etsi.org/01903#SignedProperties".  
 1090           f. Compute the digest of the resulting octet stream using the algorithm indicated in  
 1091           the time-stamp token and check if it is the same as the digest present there.
- 1092   2. Check for coherence in the value of the times indicated in the time-stamp tokens. All the  
 1093   time instants must be previous to the time when the verification is performed, to the time  
 1094   indicated within the SigningTime if present, and to the times indicated within the  
 1095   time-stamp tokens enclosed within all the rest of time-stamp container properties except  
 1096   IndividualDataObjectsTimeStamp.
- 1097   3. Set the <dss:Result> element as appropriate.
- 1098   Minor Error  
 1099   urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:InvalidAllData  
 1100   ObjectsTimeStamp MUST be used when the cryptographic verification signature succeeds  
 1101   but this timestamp verification fails.

### 1102   **6.2.3 Verification of xades:SigAndRefsTimeStamp including a XML** 1103   **timestamp**

- 1104   After completing steps 1 to 5 in section 6.2, the server will perform the steps listed below for  
 1105   completing the XML timestamp verification. If any one of them results in failure, then the  
 1106   timestamp token SHOULD be rejected.
- 1107   1. Check that those elements that, according to [XAdES] MUST be present for being  
 1108   timestamped by this timestamp, are actually present (see [XAdES] section 7.5.1).
- 1109   2. Take the other ds:Reference element and proceed to dereference it as indicated  
 1110   below:
- 1111       a. Take the XAdES elements listed in [XAdES] section 7.5.1.1 in the order indicated  
 1112       there.
- 1113       b. Canonicalize them and concatenate the resulting bytes in one octet stream. If the  
 1114       CanonicalizationMethod element of the property is present, use it for  
 1115       canonicalizing. Otherwise, use the standard canonicalization method as specified  
 1116       by [XMLSig].
- 1117       c. Compute the digest of the resulting octet stream using the algorithm indicated in  
 1118       the time-stamp token and check if it is the same as the digest present there.
- 1119   3. Check that the time indicated by the timestamp is posterior to the one indicated in the  
 1120   xades:SigningTime property, and to the times indicated in the timestamps contained  
 1121   within xades:AllDataObjectsTimeStamp,  
 1122   xades:IndividualDataObjectsTimeStamp or xades:SignatureTimeStamp, if  
 1123   present. They must also be previous to the times indicated in the timestamps enclosed by  
 1124   any xades:ArchiveTimeStamp present elements
- 1125   Minor Error  
 1126   urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:InvalidSigAndR

1127 `efsTimestamp` MUST be used when the cryptographic verification signature succeeds but  
1128 this timestamp verification fails.

#### 1129 **6.2.4 Verification of `xades:RefsOnlyTimeStamp` including a XML** 1130 **timestamp**

1131 After completing steps 1 to 5 in section 6.2, the server will perform the steps listed below for  
1132 completing the XML timestamp verification. If any one of them results in failure, then the  
1133 timestamp token SHOULD be rejected.

- 1134 1. Check that those elements that, according to [XAdES] MUST be present for being  
1135 timestamped by this timestamp, are actually present (see [XAdES] section 7.5.2).
- 1136 2. Take the other `ds:Reference` element and proceed to dereference it as indicated  
1137 below:
  - 1138 a. Take the XAdES elements listed in [XAdES] section 7.5.2.1 in the order indicated  
1139 there.
  - 1140 b. Canonicalize them and concatenate the resulting bytes in one octet stream. If the  
1141 `CanonicalizationMethod` element of the property is present, use it for  
1142 canonicalizing. Otherwise, use the standard canonicalization method as specified  
1143 by [XMLSig].
  - 1144 c. Compute the digest of the resulting octet stream using the algorithm indicated in  
1145 the time-stamp token and check if it is the same as the digest present there.
- 1146 3. Check that the time indicated by the timestamp is posterior to the one indicated in the  
1147 `xades:SigningTime` property, and to the times indicated in the timestamps contained  
1148 within `xades:AllDataObjectsTimeStamp`,  
1149 `xades:IndividualDataObjectsTimeStamp` or `xades:SignatureTimeStamp`, if  
1150 present. They must also be previous to the times indicated in the timestamps enclosed by  
1151 any `xades:ArchiveTimeStamp` present elements

1152 Minor Error

1153 `urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:InvalidRefsOnl`  
1154 `yTimeStamp` MUST be used when the cryptographic verification signature succeeds but this  
1155 timestamp verification fails.

#### 1156 **6.2.5 Verification of `xades:ArchiveTimeStamp` including a XML** 1157 **timestamp**

1158 After completing steps 1 to 5 in section 6.2, the server will perform the steps listed below for  
1159 completing the XML timestamp verification. If any one of them results in failure, then the  
1160 timestamp token SHOULD be rejected.

- 1161 1. Check that those elements that, according to [XAdES] MUST be present for being  
1162 timestamped by this timestamp, are actually present (see [XAdES] section 7.7.1).
- 1163 2. Take the other `ds:Reference` element and proceed to dereference it as indicated  
1164 below:
  - 1165 a. Take the XAdES elements listed in [XAdES] section 7.7.1 in the order indicated  
1166 there.
  - 1167 b. Canonicalize them and concatenate the resulting bytes in one octet stream. If the  
1168 `CanonicalizationMethod` element of the property is present, use it for  
1169 canonicalizing. Otherwise, use the standard canonicalization method as specified  
1170 by [XMLSig].

1171           c. Compute the digest of the resulting octet stream using the algorithm indicated in  
1172           the time-stamp token and check if it is the same as the digest present there.

1173 3. Check that the time indicated by the timestamp is posterior to the one indicated in the  
1174     SigningTime property, and to the times indicated in the timestamps contained within  
1175     xades:AllDataObjectsTimeStamp, xades:IndividualDataObjectsTimeStamp,  
1176     xades:SignatureTimeStamp if present, and xades:RefsOnlyTimeStamp or  
1177     xades:SigAndRefsTimeStamp, if present They must also be previous to the times  
1178     indicated in the timestamps enclosed by any xades:ArchiveTimeStamp that appear  
1179     before the one that is being verified

1180 Minor Error  
1181 urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:InvalidArchive  
1182 Timestamp MUST be used when the cryptographic verification signature succeeds but this  
1183 timestamp verification fails.

---

## 7 Identifiers defined in this specification

### 7.1 Predefined advanced electronic signature forms identifiers

The table below shows the URIs for standard forms of advanced electronic signature:

Advanced signature FORM	URI
XAdES-BES CAdES-BES	urn:oasis:names:tc:dss:1.0:profiles:AdES:forms:BES
XAdES-EPES CAdES-EPES	urn:oasis:names:tc:dss:1.0:profiles:AdES:forms:EPES
XAdES-T CAdES-ES-T	urn:oasis:names:tc:dss:1.0:profiles:AdES:forms:ES-T
XAdES-C CAdES-ES-C	urn:oasis:names:tc:dss:1.0:profiles:AdES:forms:ES-C
XAdES-X CAdES-ES-X	urn:oasis:names:tc:dss:1.0:profiles:AdES:forms:ES-X
XAdES-X-L CAdES--X-L	urn:oasis:names:tc:dss:1.0:profiles:AdES:forms:ES-X-L
XAdES-A CAdES-X-A	urn:oasis:names:tc:dss:1.0:profiles:AdES:forms:ES-A

Table 1.

### 7.2 Result Identifiers

This profile defines the <ResultMinor> values listed below. All of them indicate that the cryptographic verification of the signature succeeded, and that the verification of the indicated timestamp failed.

urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:InvalidIndividualDataObjectsTimestamp

urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:InvalidAllDataObjectsTimestamp

1199 urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:InvalidSigAndR  
1200 efsTimestamp  
1201 urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:InvalidRefsOnl  
1202 yTimestamp  
1203 urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:InvalidArchive  
1204 Timestamp

---

## A. Acknowledgements

1205

1206 The following individuals have participated in the creation of this specification and are  
1207 gratefully acknowledged:

1208 **Participants:**

1209 Nick Pope, Thales eSecurity  
1210 Ed Shallow, Universal Post Union  
1211 Trevor Perrin, individual

1212