



Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES); Part 1: Core Specification

STABLE DRAFT FOR PUBLIC REVIEW UNTIL 15 JANUARY 2014

Download the template for comments:

[http://docbox.etsi.org/ESI/Open/Latest Drafts/Template-for-comments.doc](http://docbox.etsi.org/ESI/Open/Latest%20Drafts/Template-for-comments.doc)

Send comments to E-SIGNATURES_COMMENTS@LIST.ETSI.ORG

CAUTION: This **DRAFT document** is provided for information and is for future development work within the ETSI Technical Committee ESI only. ETSI and its Members accept no liability for any further use/implementation of this Specification.

Approved and published specifications and reports shall be obtained exclusively via the ETSI Documentation Service at

<http://pda.etsi.org/pda/queryform.asp>

0
1
2

Reference

DEN/ESI-0019132-1

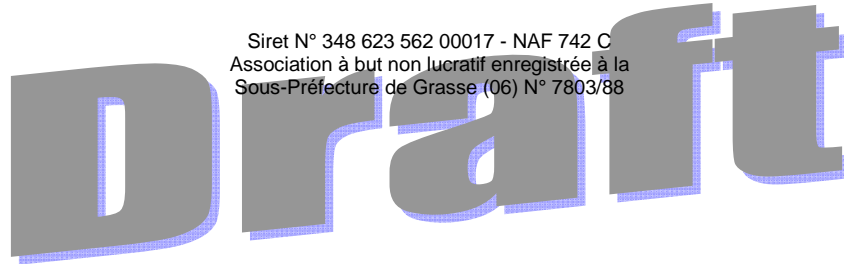
Keywords

<keywords>

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:

<http://www.etsi.org>

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at

<http://portal.etsi.org/tb/status/status.asp>

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

http://portal.etsi.org/chaicor/ETSI_support.asp

Copyright Notification

Reproduction is only permitted for the purpose of standardization work undertaken within ETSI.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2013.
All rights reserved.

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

Contents

3	Contents	3
4	Intellectual Property Rights	5
5	Foreword.....	5
6	Introduction	6
7	1 Scope	7
8	2 References	8
9	2.1 Normative references.....	8
10	2.2 Informative references.....	9
11	3 Definitions, symbols and abbreviations	9
12	3.1 Abbreviations	9
13	4 Overview	10
14	4.1 Electronic signature forms.....	12
15	4.1.1 Basic electronic signature (XAdES-BES)	12
16	4.1.2 Explicit policy electronic signatures (XAdES-EPES)	15
17	4.1.3 Electronic signature formats with validation data	16
18	4.1.3.1 Electronic signature with time (XAdES-T)	16
19	4.1.3.2 Archival electronic signatures (XAdES-A)	17
20	5 General Syntax.....	18
21	5.1 XML Namespaces for the present document.....	18
22	5.2 The QualifyingProperties element.....	19
23	5.2.1 The SignedProperties element.....	20
24	5.2.2 The UnsignedProperties element.....	20
25	5.2.3 The SignedSignatureProperties element.....	20
26	5.2.4 The SignedDataObjectProperties element.....	21
27	5.2.5 The UnsignedSignatureProperties element.....	21
28	5.2.6 The UnsignedDataObjectProperties element.....	22
29	5.3 Incorporating qualifying properties into an XML signature.....	23
30	5.3.1 Signing properties.....	23
31	5.3.2 The QualifyingPropertiesReference element.....	24
32	5.4 Managing canonicalization of XML nodesets	24
33	6 Qualifying properties syntax	24
34	6.1 Auxiliary syntax	25
35	6.1.1 The AnyType data type.....	25
36	6.1.2 The ObjectIdentifierType data type.....	25
37	6.1.3 The EncapsulatedPKIDataType data type.....	26
38	6.1.4 Types for time-stamp tokens management	27
39	6.1.4.1 Time-stamp properties in XAdES	27
40	6.1.4.2 The GenericTimeStampType data type.....	27
41	6.1.4.3 The XAdESTimeStampType data type	28
42	6.1.4.3.1 Include mechanism.....	29
43	6.1.4.4 The OtherTimeStampType data type	30
44	6.2 Properties for XAdES-BES and XAdES-EPES forms	30
45	6.2.1 The SigningTime element.....	30
46	6.2.2 References to the signing certificate.....	31
47	6.2.2.1 The SigningCertificate element	31
48	6.2.2.2 The xadesenv111:SigningCertificate element.....	32
49	6.2.3 The CommitmentTypeIndication element.....	32
50	6.2.4 The DataObjectFormat element.....	33
51	6.2.5 The SignatureProductionPlace element.....	34
52	6.2.6 Elements for incorporating signer attributes.....	34
53		

54	6.2.6.1	The SignerRole element	35
55	6.2.6.2	The xadesenv111:SignerRole element.....	35
56	6.2.7	Countersignatures	36
57	6.2.7.1	Countersignature identifier in Type attribute of ds:Reference	36
58	6.2.7.2	Enveloped countersignatures: the CounterSignature element	36
59	6.2.8	Time-stamps on signed data objects	38
60	6.2.8.1	The AllDataObjectsTimeStamp element	38
61	6.2.8.2	The IndividualDataObjectsTimeStamp element.....	38
62	6.2.9	The SignaturePolicyIdentifier element (XAdES-EPES)	38
63	6.2.9.1	Signature policy qualifier	40
64	6.2.10	The xadesenv111:SignaturePolicyStore element	40
65	6.3	The SignatureTimeStamp element (XAdES-T).....	41
66	6.4	Properties for validation data values.....	41
67	6.4.1	The CertificateValues Property element	41
68	6.4.2	The RevocationValues property element.....	42
69	6.4.3	The AttrAuthoritiesCertValues element.....	43
70	6.4.4	The AttributeRevocationValues Property element	43
71	6.5	Properties for XAdES-A form.....	44
72	6.5.1	The xadesv141:TimeStampValidationData element.....	44
73	6.5.1.1	Use of URI attribute	44
74	6.5.2	The xadesv141:ArchiveTimeStamp element	45
75	6.5.2.1	Not distributed case	45
76	6.5.2.2	Distributed case	46
77	6.5.3	The xadesenv111:RenewedDigests element	47
78	7.	Conformance requirements	50
79	7.1	XAdES-Basic Electronic Signature (XAdES-BES) conformance level.....	50
80	7.2	XAdES-Explicitly Policy based Electronic Signature (XAdES-EPES) conformance level.....	50
81	7.3	XAdES with trusted Time(XAdES-T) conformance level	51
82	7.4	XAdES with Archive-time-stamp (XAdES-A) conformance level.....	51
83		Annex <A> (normative): Additional Qualifying Properties Specification	52
84	A.1	Qualifying properties for validation data	52
85	A.1.1	References to CA certificates	52
86	A.1.1.1	The CompleteCertificateRefs element.....	52
87	A.1.1.2	The xadesenv111:CompleteCertificateRefs element.....	53
88	A.1.2	The CompleteRevocationRefs element.....	53
89	A.1.3	References to certificates in the certification path of Attribute Authorities certificates	55
90	A.1.3.1	The AttributeCertificateRefs element.....	55
91	A.1.3.2	The xadesenv111:AttributeCertificateRefs element.....	56
92	A.1.4	The AttributeRevocationRefs element.....	56
93	A.1.5	Time-stamps on references to validation data.....	57
94	A.1.5.1	The SigAndRefsTimeStamp element	57
95	A.1.5.1.1	Not distributed case	57
96	A.1.5.1.2	Distributed case	58
97	A.1.5.2	The RefsOnlyTimeStamp element.....	58
98	A.1.5.2.1	Not distributed case	59
99	A.1.5.2.2	Distributed case	59
100	A.2	Obsoleted qualifying properties	60
101	A.2.1	The ArchiveTimeStamp element	60
102	A.2.1.1	Not distributed case	60
103	A.2.1.2	Distributed case	61
104		Annex (normative): XAdES signature forms with references.....	63
105	B.1	Electronic signature with complete validation data references (XAdES-C)	63
106	B.2	Extended signatures with time forms (XAdES-X)	64
107	B.2.1	EXtended Electronic Signature with Time Type 1 (XAdES-X Type 1).....	64
108	B.2.2	EXtended Electronic Signature with Time Type 2 (XAdES-X Type 2).....	65

109	B.3	Extended long electronic signatures with time forms (XAdES-X-L type 1 or 2)	67
110	B.4	Archival Electronic Signature complete	69
111	Annex <C> (normative): Conformance requirements for additional Electronic Signature Forms.....		70
112	C.1	Electronic signatures with Complete validation data references (XAdES-C).....	70
113	C.2	EXtended signatures with time forms (XAdES-X).....	71
114	C.2.1	XAdES-X Type 1.....	71
115	C.2.2	XAdES-X Type 2.....	71
116	C.3	EXtended Long signatures with time forms (XAdES-X-L).....	71
117	C.3.1	EXtended Long signatures with time forms, Type 1 (XAdES-X-L Type 1)	71
118	C.3.2.1	EXtended Long signatures with time forms, Type 2 (XAdES-X-L Type 2)	71
119	Annex <D> (informative): General Description		72
120	D.1	The SigningTime element	72
121	D.2	References to the signing certificate	72
122	D.3	The CommitmentTypeIndication element	72
123	D.4	The DataObjectFormat element	73
124	D.5	The SignatureProductionPlace element	73
125	D.6	The SignerRole element.....	73
126	D.6.1	Claimed signer attribute/role	73
127	D.6.2	Certified signer attribute/role.....	73
128	D.7	Multiple signatures and countersignatures.....	74
129	D.8	Time-stamps on the signed data objects	75
130	D.9	The SignaturePolicyIdentifier element	75
131	D.10	The SignatureTimeStamp element	75
132	D.11	Elements containing references to validation data.....	76
133	D.12	Time-stamps on references to validation data.....	76
134	D.13	Validation data.....	77
135	D.13	Time-stamps for archival.....	78
136	D.14	The xadesenv111:RenewedDigests element.....	78
137	Annex E (informative): Change History.....		83
138	History		84
139			
140			

141 Intellectual Property Rights

142 IPRs essential or potentially essential to the present document may have been declared to ETSI. The information
 143 pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found
 144 in ETSI SR 000 314: "*Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in*
 145 *respect of ETSI standards*", which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web
 146 server (<http://ipr.etsi.org>).

147 Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee
 148 can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web
 149 server) which are, or may be, or may become, essential to the present document.

150 Foreword

151 This draft European Standard (EN) has been produced by ETSI Technical Committee Electronic Signatures and
 152 Infrastructures (ESI) and is now submitted for getting public comments from stakeholders.

153 The present document is part 1 of a multi-part deliverable covering the XML Advanced Electronic Signatures (XAdES),
 154 as identified below:
 155

156 **Part 1: Core Specification**

157 Part 2: XAdES Baseline Profile

Proposed national transposition dates	
Date of latest announcement of this EN (doa):	3 months after ETSI publication
Date of latest publication of new National Standard or endorsement of this EN (dop/e):	6 months after doa
Date of withdrawal of any conflicting National Standard (dow):	6 months after doa

158

159 Introduction

160 Electronic commerce has emerged as a frequent way of doing business between companies across local, wide area and
 161 global networks. Trust in this way of doing business is essential for the success and continued development of
 162 electronic commerce. It is therefore important that companies using this electronic means of doing business have
 163 suitable security controls and mechanisms in place to protect their transactions and to ensure trust and confidence with
 164 their business partners. In this respect the electronic signature is an important security component that can be used to
 165 protect information and provide trust in electronic business.

166 The European Directive on a community framework for Electronic Signatures (also denoted as "the Directive" or the
 167 "European Directive" in the rest of the present document) defines an electronic signature as: "data in electronic form
 168 which is attached to or logically associated with other electronic data and which serves as a method of authentication".

169 The present document is intended to cover electronic signatures for various types of transactions, including business
 170 transactions (e.g. purchase requisition, contract and invoice applications). Thus the present document can be used for
 171 any transaction between an individual and a company, between two companies, between an individual and a
 172 governmental body, etc. The present document is independent of any environment. It can be applied to any environment
 173 e.g. smart cards, GSM SIM cards, special programs for electronic signatures, etc.

174 The present document:

- 175 • specifies XML schema [4] definitions for new XML types that can be used to generate properties that further
 176 qualify XMLDSIG signatures with information able to fulfil a number of common requirements such as the
 177 long term validity of the signature by usage of time-stamps, etc.;
- 178 • defines mechanisms for incorporating the aforementioned qualifying information;
- 179 • specifies formats for XML advanced electronic signatures that, by using the specified new XML types, remain
 180 valid over long periods and incorporate additional useful information in common use cases. These signatures
 181 will be built on XMLDSIG by encapsulating these properties (and/or references to them) within one
 182 `ds:Object` XML element defined in [2]. Here, as for the rest of the document, `ds` has been used as the
 183 prefix denoting the namespace whose URI value is "http://www.w3.org/2000/09/xmlsig#";
- 184 • defines a set of conformance requirements to claim endorsement to the present document.

185 The present document specifies two main types of properties: signed properties and unsigned properties. The first ones
 186 are additional data objects that are also secured by the signature produced by the signer on the `ds:SignedInfo`
 187 element, which implies that the signer gets these data objects, computes a hash for all of them and generates the
 188 corresponding `ds:Reference` element. The unsigned properties are data objects added by the signer, by the verifier
 189 or by other parties after the production of the signature. They are not secured by the signature in the `ds:Signature`
 190 element (the one computed by the signer); however they can be actually signed by other parties (time-stamps,
 191 countersignatures, certificates and CRLs are also signed data objects).

192 **EDITOR NOTE:** a number of editor notes like this one appears throughout the document. These notes intend to
 193 attract readers' attention and/or kindly request their feedback on certain specific issues.

194 EDITOR NOTE: ESI is assessing whether it is suitable or not to incorporate, in a normative annex, part of the
195 material present in ETSI TS 101 903 v1.4.2, annex G: “Details on XAdES signatures validation”. ESI has
196 issued ETSI TS 102 853: “Signature verification procedures and policies”, and will issue, based on this
197 document, the future ETSI EN 319 102: “Procedures for signature creation and validation”. It is the
198 intention that this last document presents a validation procedures without entering in issues specific to the
199 formats. In consequence, ESI is assessing if relevant validation issues specific to XAdES format would be
200 missed if this annex is completely suppressed. Feedback from stakeholders regarding the suitability of
201 keeping parts of such an annex would be highly appreciated.

202 1 Scope

203 The present document defines XML [6] formats for advanced electronic signatures that remain valid over long periods,
204 are compliant with the European Directive and incorporate additional useful information in common uses cases. This
205 includes evidence as to its validity even if the signer or verifying party later attempts to deny (repudiates) the validity of
206 the signature.

207 The present document is based on the use of public key cryptography to produce digital signatures, supported by public
208 key certificates.

209 The present document uses a signature policy, implicitly or explicitly referenced by the signer, as one possible basis for
210 establishing the validity of an electronic signature.

211 The present document uses time-stamps or trusted records (e.g. time-marks) to prove the validity of a signature long
212 after the normal lifetime of critical elements of an electronic signature and to support non-repudiation. It also specifies
213 the optional use of additional time-stamps to provide very long-term protection against key compromise or weakened
214 algorithms.

215 The present document then, specifies the use of the corresponding trusted service providers (e.g. time-stamping
216 authorities), and the data that needs to be archived (e.g. cross certificates and revocation lists).

217 An advanced electronic signature aligned with the present document can, in consequence, be used for arbitration in case
218 of a dispute between the signer and verifier, which may occur at some later time, even years later.

219 The present document:

- 220 • shows a taxonomy of the qualifying information (properties) whose presence in an electronic signature allows
221 it to remain valid over long periods, to satisfy common use cases requirements, and to be compliant with the
222 European Directive;
- 223 • specifies XML schema definitions for new elements able to carry or to refer to the aforementioned properties;
- 224 • specifies two ways for incorporating the qualifying information to XMLDSIG, namely either by direct
225 incorporation of the qualifying information or using references to such information. Both ways make use of
226 mechanisms defined in XMLDSIG.

227 Clause 4 gives an overview of some of the various types of advanced electronic signatures defined in the present
228 document.

229 Clause 5 defines the namespaces used in the XML schema definitions appearing in the present document. It also defines
230 the types for the containers of the qualifying properties, and specifies the mechanisms for incorporating them to the
231 electronic signature.

232 Clause 6 defines the qualifying properties except those properties that contain references to validation data, time-stamps
233 on these properties and a former container for archive time-stamps, nowadays superseded but kept for the sake of legacy
234 XAdES signatures .

235 Clause 7 defines different conformance levels that may be claimed against the present document.

236 Normative Annex A defines qualifying properties that encapsulate references to validation data, properties that
237 encapsulate time-stamp tokens for these references, and properties that have been obsoleted by new properties (for
238 preserving management of legacy electronic signatures).

239 Normative Annex B defines XAdES forms that incorporate properties encapsulating references to validation data and
240 properties encapsulating time-stamp tokens for these references.

241 Normative Annex C specifies conformance levels for XAdES signatures that incorporate properties encapsulating
242 references to validation data and properties encapsulating time-stamp tokens for these references.

243 Informative Annex D provides rationale for the properties and mechanisms specified by this document.

244 Informative Annex E provides some information of the main changes introduced in the present document with regards
245 to ETSI TS 101 903 [i.2].

246 2 References

247 References are either specific (identified by date of publication and/or edition number or version number) or
248 non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the
249 referenced document (including any amendments) applies.

250 Referenced documents which are not found to be publicly available in the expected location might be found at
251 <http://docbox.etsi.org/Reference>.

252 NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee
253 their long term validity.

254 2.1 Normative references

255 The following referenced documents are necessary for the application of the present document.

- 256 [1] ETSI TS 101 903: "Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic
257 Signatures (XAdES)".
- 258 [2] W3C Recommendation: "XML-Signature Syntax and Processing, Version 1.1".
- 259 [3] W3C Recommendation Part 1 (28 October 2004): "XML Schema Part 1: Structures".
- 260 [4] W3C Recommendation Part 2 (28 October 2004): "XML Schema Part 2: Datatypes".
- 261 [5] ITU-T Recommendation X.509: "Information technology - Open Systems Interconnection - The
262 Directory: Public-key and attribute certificate frameworks".
- 263 [6] W3C Recommendation (26 November 2008): "Extensible Markup Language (XML) 1.0".
- 264 [7] IETF RFC 2560: "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol -
265 OCSP".
- 266 [8] IETF RFC 3161: "Internet X.509 Public Key Infrastructure Time Stamp Protocol (TSP)".
- 267 [9] IETF RFC 5280: "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation
268 List (CRL) Profile".
- 269 [10] IETF RFC 3061: "A URN Namespace of Object Identifiers".
- 270 [11] ETSI EN 319 102: "Electronic Signatures and Infrastructures (ESI); Procedures for Signature
271 Creation and Validation".
- 272 [12] ETSI EN 319 172: "Electronic Signatures and Infrastructures (ESI); Signature Policies".
- 273 [13] IETF RFC 6931: "Additional XML Security Uniform Resource Identifiers (URIs)".

274

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area

- [i.1] ETSI EN 319 411-4: "Electronic Signatures and Infrastructures (ESI); Policy and security requirements for Trust Service Providers issuing certificates; Part 4: Policy requirements for certification authorities issuing attribute certificates".
- [i.2] ETSI TS 101 903 v1.4.2: "Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signature (XAdES)"

3 Definitions, symbols and abbreviations

3.1 Abbreviations

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

286	AARL	Attribute Authority Revocation List
287	AC	Attribute Certificate
288	ASN.1	Abstract Syntax Notation 1
289	BER	Basic Encoding Rules
290	CA	Certification Authority
291	CER	Canonical Encoding Rules
292	CMS	Cryptographic Message Syntax
293	CRL	Certificate Revocation List
294	DER	Distinguished Encoding Rules
295	DTD	Document Type Definition
296	ES	Electronic Signature
297	HTTP	Hyper Text Transfer Protocol
298	OCSP	Online Certificate Status Protocol
299	OID	Object Identifier
300	PER	Packed Encoding Rules
301	PKC	Public Key Certificate
302	TSA	Time-Stamping Authorities
303	TSP	Trusted Service Providers
304	TSU	Time Stamping Unit
305	URI	Uniform Resource Identifier
306	URN	Uniform Resource Name
307	XAdES	XML Advanced Electronic Signature
308	XAdES-A	XAdES Archiving validation data
309	XAdES-BES	XAdES Basic Electronic Signature
310	XAdES-C	XAdES Complete validation data
311	XAdES-EPES	XAdES Explicit Policy based Electronic Signature
312	XAdES-T	XAdES with Time-stamp
313	XAdES-X	XAdES eXtended validation data
314	XER	XML Encoding Rules
315	XML	eXtensible Markup Language
316	XMLDSIG	eXtensible Markup Language Digital SIGNature
317	XSLT	eXtensible Stylesheet Language Transformations

4 Overview

The present document defines a set of signature properties that may be combined to obtain electronic signature forms providing satisfaction of different requirements. Below follows a short overview of the properties:

- `SigningCertificate` and `xadesenv111:SigningCertificate`. These properties contain an unambiguous reference to the signer's certificate, formed by its identifier and the digest value of the certificate. Its usage is particularly important when a signer holds a number of different certificates containing the same public key, to avoid claims by a verifier that the signature implies another certificate with different semantics. This is also important when the signer holds different certificates containing different public keys in order to provide the verifier with the correct signature validation data. Finally, it is also important in case the issuing key of the CA providing the certificate would be compromised. Details on these properties can be found in clause 6.2.2.
 - `SignaturePolicyIdentifier`. This property contains information being an **unambiguous way for identifying the signature policy** under which the electronic signature has been produced. This will ensure that the verifier will be able to use the same signature policy during the validation process. A signature policy is useful to clarify the precise role and commitments that the signer intends to assume with respect to the signed data object, and to avoid claims by the verifier that the signer implied a different signature policy. Details on this property can be found in clause 6.2.9.
 - Validation data properties. The present document defines a number of XML types able to contain both validation data (certificate chains, CRLs, OCSP responses, etc.) and references to them (identifiers of certificates, CRLs, OCSP responses, etc.). Properties of these types allow to incorporate all material used for validation into the signature. They can be jointly used with time-stamp properties to provide long-term validity. Some of them contain validation data:
 - `CertificateValues`. It contains the values of certificates used to validate the signature. Details on this property can be found in clause 6.4.1.
 - `RevocationValues`. It contains revocation information used for the validation of the electronic signature. Details on this property can be found in clause 6.4.2.
 - `AttrAuthoritiesCertValues`. It contains values of the Attribute Authorities certificates that have been used to validate the attribute certificate when present in the signature. It may also contain CA certificates in the certification path of the Attribute Authorities certificates. It may also contain certificates from entities issuing signed assertions on the signer, and CA certificates within their certification paths. Details on this property can be found in clause 6.4.3.
 - `AttributeRevocationValues`. It contains the full set of revocation data that have been used to validate the attribute certificate when present in the signature. It may also contain the full set of revocation data that have been used to validate signed assertions on the signer. Details on this property can be found in clause 6.4.4.
- Other contain references to validation data:
- `CompleteCertificateRefs` and `xadesenv111:CompleteCertificateRefs`. References to the CA certificates used to validate the signature. Details on these properties can be found in normative annex A clause A.1.1.
 - `CompleteRevocationRefs`. It contains references to the full set of revocation information used for the validation of the electronic signature. Details on this property can be found in normative annex A clause A.1.2.
 - `AttributeCertificateRefs` and `xadesenv111:AttributeCertificateRefs`. They contain references to the full set of Attribute Authorities certificates that have been used to validate the attribute certificate. It may also contain references to CA certificates in the certification path of the Attribute Authorities certificates. It may also contain references to certificates from entities issuing signed assertions on the signer, and references to CA certificates within their certification paths. Details on these properties can be found in normative annex A clause A.1.3.

- 366 - `AttributeRevocationRefs`. It contains references to the full set of references to the revocation
367 data that have been used in the validation of the attribute certificate(s) present in the signature. It may
368 also contain the full set of references to revocation data that have been used to validate signed assertions
369 on the signer. Details on this property can be found in normative annex A clause A.1.4.
- 370
- 371 • Time-stamp token container properties. The present document defines an abstract and two concrete XML
372 types (`GenericTimeStampType`, `XAdESTimeStampType` and `OtherTimeStampType`) for allowing
373 the inclusion of time-stamp tokens in a XMLDSIG signature. These types are defined in clause 6.1.4. The
374 present document uses `XAdESTimeStampType` for defining several time-stamp token container properties,
375 with capability for containing one or more time-stamp tokens covering different parts of the signature
376 (common elements defined in XMLDSIG, validation data, qualifying properties, etc.). It also specifies a type
377 for encapsulating validation data used for validating time-stamp tokens. Below follows the list:
- 378 - `SignatureTimeStamp`. Each time-stamp token within this property covers the digital signature value
379 element. Details on this property can be found in clause 6.3.
- 380 - `AllDataObjectsTimeStamp`. Each time-stamp token within this property covers all the signed data
381 objects. Details on this property can be found in clause 6.2.8.1.
- 382 - `IndividualDataObjectsTimeStamp`. Each time-stamp token within this property covers selected
383 signed data objects. Details on this property can be found in clause 6.2.8.2.
- 384 - `xadesv141:ArchiveTimeStamp`. Each time-stamp token within this property covers signature and
385 other properties required for achieving an Archival Electronic Signature and for providing long-term
386 validity. Details on this property can be found in clause 6.5.2.
- 387 - `SigAndRefsTimeStamp`. Each time-stamp token within this property covers the signature and
388 references to validation data. Details on this property can be found in normative annex A clause A.1.5.1.
- 389 - `RefsOnlyTimeStamp`. Each time-stamp token within this property covers only references to
390 validation data. Details on this property can be found in annex A clause A.1.5.2.
- 391 - `ArchiveTimeStamp`. This element is deprecated by `xadesv141:ArchiveTimeStamp` and kept
392 within the normative annex A for managing legacy XAdES signatures. Details on this property can be
393 found in normative annex A clause A.2.1.
- 394 - `xadesv141:TimeStampValidationData`. This property encapsulates validation data for a time-
395 stamp token embedded in one of the XAdES time-stamp token containers. Details on this property can be
396 found in clause 6.5.1.
- 397 • Other properties. The present document defines a number of additional properties that can be useful in a wide
398 range of environments, namely:
- 399 - `SigningTime`. This property contains the time at which the signer claims to have performed the
400 signing process. Details on this property can be found in clause 6.2.1.
- 401 - `DataObjectFormat`. This property identifies the format of a signed data object (when electronic
402 signatures are not exchanged in a restricted context) to enable the presentation to the verifier or the use
403 by the verifier (text, sound or video) in exactly the same way as intended by the signer. Details on this
404 property can be found in clause 6.2.4.
- 405 - `CommitmentTypeIndication`. This property identifies the commitment undertaken by the signer in
406 signing (a) signed data object(s) in the context of the selected signature policy (when an explicit
407 commitment is being used). This will be required where a signature policy specifies more than a single
408 commitment type, each of which might have different legal interpretations of the intent of the signature
409 (e.g. proof of origin, proof of receipt, proof of creation, etc.). Details on this property can be found in
410 clause 6.2.3.
- 411 - `SignatureProductionPlace`. This property contains the indication of the purported place where
412 the signer claims to have produced the signature. Details on this property can be found in clause 6.2.5.

- 413 - `SignerRole` and `xadesenv111:SignerRole`. These properties allow incorporating signer
414 attributes (e.g. signer roles). `SignerRole`, specified in clause 6.2.6.1 allows to incorporate claimed or
415 certified attributes using X509 based attribute certificates, issued by an Attribute Authority. Property
416 `xadesenv111:SignerRole`, which is specified in clause 6.2.6.2 allows incorporating the
417 aforementioned types of attributes and also signed attributes in XML syntax (for instance signed SAML
418 assertions).
- 419 - `CounterSignature`. This property contains signature(s) produced on the signature. Details on this
420 property can be found in clause 6.2.7.2.
- 421 - `xadesenv111:SignaturePolicyStore`. This property allows incorporating to the signature the
422 signature policy document or a pointer to a local storage where this document may be stored for
423 managing the signature in the long term. Details on this property may be found in clause 6.2.10.
- 424 - `xadesenv111:RenewedDigests`. This property contains digest values of detached data objects
425 indirectly signed by using signed `ds:Manifest` elements. It allows to counter a threat resulting from
426 the combination of the break of some of the digest algorithms used within the aforementioned signed
427 `ds:Manifest` and the substitution of some detached data objects. Clause 6.5.3 specifies this property.
428 Clause D1.14 provides the rationale for this property.

429 The aforementioned properties may be combined to generate different electronic signature forms. Some of them are
430 defined in clause 4.1 of its normative part. Additional extended forms are defined in the normative annex B.

431 4.1 Electronic signature forms

432 The current clause specifies four forms of XML advanced electronic signatures, namely the **Basic Electronic**
433 **Signature** (XAdES-BES), the **Explicit Policy based Electronic Signature** (XAdES-EPES), the **Electronic Signature**
434 **with Time** (XAdES-T), and the **Archival Electronic Signature** (XAdES-A).

435 The normative annex B defines forms of XAdES signatures incorporating properties that encapsulate references to
436 validation data and properties that encapsulate time-stamp tokens on the aforementioned references.

437 Readers are referred to RFC 6931 [13] where they may find additional URIs to those ones defined within XMLDSIG
438 [2].

439 4.1.1 Basic electronic signature (XAdES-BES)

440 A Basic Electronic Signature (XAdES-BES) in accordance with the present document will build on a XMLDSIG by
441 incorporating qualifying properties defined in the present document. They will be incorporated to XMLDSIG using one
442 of the mechanisms specified in clause 5.3.

443 Some properties defined for building up this form will be covered by the signer's signature (descendant of
444 `SignedProperties` element). Other properties will be not covered by the signer's signature (descendant of
445 `UnsignedProperties` element).

446 In a XAdES-BES the signature value shall be computed in the usual way of XMLDSIG over the data object(s) to be
447 signed and on the whole set of signed properties when present (`SignedProperties` element).

448 For this form it is mandatory to protect the signing certificate with the signature, in one of the two following ways:

- 449 • either incorporating one of the properties referencing the signing certificate (i.e. `SigningCertificate` or
450 `xadesenv111:SigningCertificate` signed properties); or
- 451 • not incorporating none of the aforementioned signed properties but incorporating the signing certificate within
452 the `ds:KeyInfo` element and signing at least the signing certificate.

453 A XAdES-BES signature shall, in consequence, contain at least one of the following elements with the specified
454 contents:

- 455 • One of the `SigningCertificate` or the `xadesenv111:SigningCertificate` signed properties.
456 The present property shall contain the reference and the digest value of the signing certificate. It may contain
457 references and digests values of other certificates (that may form a chain up to the point of trust). In the case of
458 ambiguities identifying the actual signer's certificate the applications should include one of these properties.
- 459 • The `ds:KeyInfo` element. If one of `SigningCertificate` or
460 `xadesenv111:SigningCertificate` is incorporated to the signature, no restrictions apply to this
461 element. If none of them are incorporated to the signature, then the following restrictions apply:
 - 462 - the `ds:KeyInfo` element shall include a `ds:X509Data` containing the signing certificate;
 - 463 - the `ds:KeyInfo` element also may contain other certificates forming a chain that MAY reach the point
464 of trust;
 - 465 - the `ds:SignedInfo` element shall contain a `ds:Reference` element referencing `ds:KeyInfo`.
466 That `ds:Reference` element shall be built in such a way that at least the signing certificate is actually
467 signed.

468 NOTE 1: Readers are warned that signing the whole `ds:KeyInfo` locks the element: any addition of a certificate
469 or validation data would make signature validation fail. Applications may, alternatively, use XPath
470 transforms for signing at least the signing certificate, leaving the `ds:KeyInfo` element open for
471 addition of new data after signing.

472 By incorporating one of these elements, XAdES-BES prevents the simple substitution of the signer's certificate
473 (see clause D.1.2).

474 A XAdES-BES signature may also contain the following properties:

- 475 • the `SigningTime` signed property;
- 476 • the `DataObjectFormat` signed property;
- 477 • the `CommitmentTypeIndication` signed property;
- 478 • the `SignerRole` or `xadesenv111:SignerRole` signed property;
- 479 • the `SignatureProductionPlace` signed property;
- 480 • one or more `IndividualDataObjectsTimeStamp` or `AllDataObjectTimeStamp` signed
481 properties;
- 482 • one or more `CounterSignature` unsigned properties.

483 Below follows the structure of the XAdES-BES built by direct incorporation of the qualifying information in the
484 corresponding new XML elements to the XMLDSIG.

485 In the examples shown within these clauses "?" denotes zero or one occurrence; "+" denotes one or more occurrences;
486 and "*" denotes zero or more occurrences. In the examples shown in these clauses, "ds" stands for the prefix
487 corresponding to the namespace whose URI value is "http://www.w3.org/2000/09/xmldsig#", and absence of prefix
488 stands for those elements defined within the namespace whose URI value is "http://uri.etsi.org/01903/v1.3.2#".
489 Component [Ref. to signing certificate] denotes the choice between `SigningCertificate` and
490 `xadesenv111:SigningCertificate`. [Signer Attrs] denotes the choice between `SignerRole` and
491 `xadesenv111:SignerRole`.

492

```

493                                     XMLDSIG
494                                     |
495 <ds:Signature ID?>- - - - -+-----+
496   <ds:SignedInfo>
497     <ds:CanonicalizationMethod/>
498     <ds:SignatureMethod/>
499     (<ds:Reference URI? >
500       (<ds:Transforms>)?
501       <ds:DigestMethod/>
502       <ds:DigestValue/>
503     </ds:Reference>)+
504   </ds:SignedInfo>
505   <ds:SignatureValue/>
506   (<ds:KeyInfo>)?- - - - -+
507
508   <ds:Object>
509
510     <QualifyingProperties?>
511
512       <SignedProperties?>
513
514         <SignedSignatureProperties>
515           (SigningTime)?
516           ([Ref.to signing certificate])?
517           (SignatureProductionPlace)?
518           ([Signer Attrs.]?)
519         </SignedSignatureProperties>
520
521         <SignedDataObjectProperties>
522           (DataObjectFormat)*
523           (CommitmentTypeIndication)*
524           (AllDataObjectsTimeStamp)*
525           (IndividualDataObjectsTimeStamp)*
526         </SignedDataObjectProperties>
527       </SignedProperties>
528
529       <UnsignedProperties?>
530
531         <UnsignedSignatureProperties?>
532           (CounterSignature)*
533         </UnsignedSignatureProperties>
534
535       </UnsignedProperties>
536
537     </QualifyingProperties>
538
539   </ds:Object>
540 </ds:Signature>- - - - -+-----+
541
542                                     XAdES-BES
543
544
545

```

546 Other XMLDSIG ds:Object elements with different contents may be added within the structure shown above to
 547 satisfy requirements other than the ones expressed in the present document. This also applies to the rest of the examples
 548 of structures of XAdES forms shown in this clause.

549 NOTE 2: The XAdES-BES is the minimum format for an electronic signature to be generated by the signer. On its
 550 own, it does not provide enough information for it to be verified in the longer term.

551 The XAdES-BES satisfies the legal requirements for electronic signatures as defined in the European Directive on
 552 electronic signatures. It provides basic authentication and integrity protection.

553 The XAdES-BES form is the XAdES instantiation of the AdES-BES form specified within ETSI EN 319 102.

554 Conformance requirements for this form of XAdES signatures are specified in clause 7.1.

555

4.1.2 Explicit policy electronic signatures (XAdES-EPES)

556

557

558

559

560

An **Explicit Policy based Electronic Signature** (XAdES-EPES) form in accordance with the present document, extends the definition of an electronic signature to conform to the identified signature policy. A XAdES-EPES builds up on a XAdES-BES forms by incorporating the `SignaturePolicyIdentifier` element. This signed property indicates that a signature policy shall be used for signature validation. It provides means for explicitly identifying the signature policy. Other properties may be required by the mandated policy.

561

562

Clause 6.2.9 provides details on the specification of `SignaturePolicyIdentifier` property. Specification of the actual signature policies is outside the scope of the present document.

563

564

The structure of the XAdES-EPES (created **by direct incorporation** of the qualifying information to a XAdES-BES form) is illustrated below.

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

```

XMLDSIG
|
<ds:Signature ID?>- - - - -+
|
|  <ds:SignedInfo>
|  <ds:CanonicalizationMethod/>
|  <ds:SignatureMethod/>
|  (<ds:Reference URI? >
|   (<ds:Transforms>)?
|   <ds:DigestMethod/>
|   <ds:DigestValue/>
|  </ds:Reference>)+
|  </ds:SignedInfo>
|  <ds:SignatureValue/>
|  (<ds:KeyInfo>)?- - - - -+
|
|  <ds:Object>
|  <QualifyingProperties>
|  <SignedProperties>
|  <SignedSignatureProperties>
|  (SigningTime)?
|  ([Ref. to signing certificate])?
|  (SignaturePolicyIdentifier)
|  (SignatureProductionPlace)?
|  ([Signer Attrs.]?)
|  </SignedSignatureProperties>
|  <SignedDataObjectProperties>
|  (DataObjectFormat)*
|  (CommitmentTypeIndication)*
|  (AllDataObjectsTimeStamp)*
|  (IndividualDataObjectsTimeStamp)*
|  </SignedDataObjectProperties>
|  </SignedProperties>
|  <UnsignedProperties?>
|  <UnsignedSignatureProperties?>
|  (CounterSignature)*
|  </UnsignedSignatureProperties>
|  </UnsignedProperties>
|  </QualifyingProperties>
|  </ds:Object>
|
+-----+
XAdES-EPES

```

618

The XAdES-EPES form is the XAdES instantiation of the AdES-EPES form specified within ETSI EN 319 102.

619

Conformance requirements for this form of XAdES signatures are specified in clause 7.2.

620

521 4.1.3 Electronic signature formats with validation data

522 Validation of an electronic signature in accordance with the present document requires additional data needed to
523 validate the electronic signature. This additional data is called **validation data**; and includes:

- 524 • Public Key Certificates (PKCs) and Attributes Certificates (ACs);
- 525 • revocation status information for each PKC and AC (for instance Certificate Revocation Lists –CRLs- or
526 OCSF responses);
- 527 • trusted time-stamps applied to the digital signature or a time-mark that shall be available in an audit log,
528 evidencing that certain objects existed before a particular point in time;
- 529 • when appropriate, the details of a signature policy to be used to verify the electronic signature.

530 The present document defines properties able to encapsulate validation data. Clauses below summarize some signature
531 forms that incorporate them and their most relevant characteristics

532 4.1.3.1 Electronic signature with time (XAdES-T)

533 XML **Advanced Electronic Signature with Time** (XAdES-T) builds on a XAdES-BES or a XAdES-EPES by
534 incorporation of a trusted time associated to the signature. The trusted time may be provided by two different means:

- 535 • the `SignatureTimeStamp` as an unsigned property added to the electronic signature;
- 536 • a time mark of the electronic signature provided by a trusted service provider.

537 A time-mark provided by a Trusted Service would have similar effect to the `SignatureTimeStamp` property but in
538 this case no property is added to the electronic signature as it is the responsibility of the TSP to provide evidence of a
539 time mark when required to do so. The management of time marks is outside the scope of the present document.

540 **EDITOR NOTE: Feedback is kindly requested on the suitability of including a referencing mechanism to a time-**
541 **mark. We are especially interested in the opinion of stakeholders using a time-mark mechanism**

542 Trusted time provides the initial steps towards providing long term validity. The XAdES-T trusted time indications shall
543 be created before a certificate has been revoked or expired.

544 Below follows the structure of a XAdES-T form built on a XAdES-BES or a XAdES-EPES (if
545 `SignaturePolicyIdentifier` signed property is present), by direct incorporation of a time-stamp token within
546 the `SignatureTimeStamp` element. A XAdES-T form based on time-marks may exist without such an element.

```

547                                     XMLDISG
548                                     |
549 <ds:Signature ID?>- - - - - + - - - - + - - - - +
550 <ds:SignedInfo> |
551   <ds:CanonicalizationMethod/> |
552   <ds:SignatureMethod/> |
553   (<ds:Reference URI? > |
554     (<ds:Transforms>)? |
555     <ds:DigestMethod/> |
556     <ds:DigestValue/> |
557   </ds:Reference>)+ |
558 </ds:SignedInfo> |
559 <ds:SignatureValue/> |
560 (<ds:KeyInfo>)? - - - - - +
561
562 <ds:Object>
563
564   <QualifyingProperties>
565
566     <SignedProperties?>
567
568       <SignedSignatureProperties>
569         (SigningTime)?
570         ([Ref. to signing certificate])?
571         (SignaturePolicyIdentifier)?
572         (SignatureProductionPlace)?
573         ([Signer Attrs.]?)
574       </SignedSignatureProperties>

```



```

575     <SignedDataObjectProperties>
576         (DataObjectFormat) *
577         (CommitmentTypeIndication) *
578         (AllDataObjectsTimeStamp) *
579         (IndividualDataObjectsTimeStamp) *
580     </SignedDataObjectProperties>
581
582 </SignedProperties>
583
584 <UnsignedProperties>
585     <UnsignedSignatureProperties>
586         (CounterSignature)* - - - - - +
587         (SignatureTimeStamp)+
588     </UnsignedSignatureProperties> - - - +
589
590 </UnsignedProperties>
591
592 </QualifyingProperties>
593
594 </ds:Object>
595
596 </ds:Signature> - - - - - +
597
598
599
600 XAdES-BES (-EPES)
601
602 XAdES-T
603

```

The XAdES-T form is the XAdES instantiation of the AdES-T form specified within ETSI EN 319 102.

Conformance requirements for this form of XAdES signatures are specified in clause 7.3.

NOTE : As a minimum, the signer will provide the XAdES-BES or when indicating that the signature conforms to an explicit signing policy the XAdES-EPES. To reduce the risk of repudiating signature creation, the trusted time indication needs to be as close as possible to the time the signature was created. The signer or a TSP could provide the XAdES-T. If the signer did not provide it, the verifier SHOULD create the XAdES-T on first receipt of an electronic signature, because the XAdES-T provides independent evidence of the existence of the signature prior to the trusted time indication.

4.1.3.2 Archival electronic signatures (XAdES-A)

Archival signatures in accordance with the present document incorporate `CertificateValues` unless the `ds:KeyInfo` element does contain the full set of certificates used to validate the electronic signature. They also incorporate `RevocationValues` unless the `ds:KeyInfo` element contains the revocation information that has to be shipped with the electronic signature. Archival signatures also incorporate one or more `xadesv141:ArchiveTimeStamp` unsigned properties. They may contain other properties. Each `xadesv141:ArchiveTimeStamp` element contains time-stamp tokens covering among other elements, those ones that contain validation data. These forms are used for archival of signatures. Successive archive time-stamps protect the whole material against vulnerable hashing algorithms or the breaking of the cryptographic material or algorithms and the expiration of the time-stamp token certificate.

Below follows the structure of a XAdES-A built on a XAdES-T by incorporation of at least one `xadesv141:ArchiveTimeStamp` element. In the figure below, the prefix "xadesv141" prefix corresponds to XML Namespace whose URI value is "<http://uri.etsi.org/01903/v1.4.1#>"

```

726
727 XMLDISG
728 |
729 <ds:Signature ID?> - - - - - +
730 <ds:SignedInfo>
731 <ds:CanonicalizationMethod/>
732 <ds:SignatureMethod/>
733 (<ds:Reference URI? >
734 (<ds:Transforms/>)?
735 <ds:DigestMethod/>
736 <ds:DigestValue/>
737 </ds:Reference>)+
738 </ds:SignedInfo>

```

```

738 <ds:SignatureValue/> |
739 (<ds:KeyInfo>)? - - - - - +
740
741 <ds:Object>
742
743   <QualifyingProperties>
744
745     <SignedProperties>?
746
747       <SignedSignatureProperties>
748         (SigningTime)?
749         ([Ref. to signing certificate])?
750         (SignaturePolicyIdentifier)?
751         (SignatureProductionPlace)?
752         ([Signer Attrs.]?)
753       </SignedSignatureProperties>
754
755       <SignedDataObjectProperties>
756         (DataObjectFormat)*
757         (CommitmentTypeIndication)*
758         (AllDataObjectsTimeStamp)*
759         (IndividualDataObjectsTimeStamp)*
760       </SignedDataObjectPropertiesSigned>
761
762     </SignedProperties>
763
764     <UnsignedProperties>
765
766       <UnsignedSignatureProperties>
767         (xadesv141:TimeStampValidationData)* + <- These elements may contain revocation
768         information of time-stamp tokens embedded
769         in AllDataObjectsTimeStamp or
770         IndividualDataObjectsTimeStamp
771
772         (CounterSignature)*
773         ((SignatureTimeStamp)
774         (xadesv141:TimeStampValidationData)?) * <- Properties containing validation material
775         (CertificatesValues)? <- are optional as this may be already present
776         (RevocationValues)? <- within ds:KeyInfo
777         (AttrAuthoritiesCertValues)?
778         (AttributeRevocationValues)?
779         ( (xadesv141:ArchiveTimeStamp
780           xadesv141:TimeStampValidationData? ) *
781         </UnsignedSignatureProperties>- - - - |
782
783     </UnsignedProperties>
784
785   </QualifyingProperties>
786
787 </ds:Object>
788 </ds:Signature>- - - - - +
789
790 XAdES-A

```

The XAdES-A form is the XAdES instantiation of the AdES-A form specified within ETSI EN 319 102.

Conformance requirements for this form of XAdES signatures are specified in clause 7.4.

5 General Syntax

The present clause defines the namespaces used in the XML schema definitions appearing in the present document. It also defines the types for the containers of the qualifying properties, and specifies the mechanisms for incorporating them to the electronic signature.

5.1 XML Namespaces for the present document

The present document uses the following URI namespaces:

- <http://uri.etsi.org/01903/v1.3.2#>

- 801 • <http://uri.etsi.org/01903/v1.4.1#>
- 802 • <http://uri.etsi.org/19132/v1.1.1#>
- 803 • <http://www.w3.org/2000/09/xmldsig#>
- 804 • <http://www.w3.org/2001/XMLSchema>

805 The table below shows the correspondence between the namespaces' URIs and the prefixes used throughout the present
806 document:

807 **Table 5.1.1: Namespaces and prefixes**

XML Namespace URI	Prefix
http://uri.etsi.org/01903/v1.3.2#	xades
http://uri.etsi.org/01903/v1.4.1#	xadesv141
http://uri.etsi.org/19132/v1.1.1#	xadesenv111
http://www.w3.org/2000/09/xmldsig#	ds
http://www.w3.org/2001/XMLSchema	xsd

808
809 NOTE 1: The <http://uri.etsi.org/01903/v1.3.2#> URI was defined by ETSI TS 101 903 v1.3.2. Most of the XML
810 elements and types used by XAdES signatures were defined in this namespace. Additionally, ETSI TS
811 101 903 v.1.4.1 defined <http://uri.etsi.org/01903/v1.4.1#> URI, where new types and elements were
812 defined. Finally, the present document will also define new types and elements, which will be defined
813 within the namespace whose URI is <http://uri.etsi.org/19132/v1.1.1#>.

814 NOTE 2: This specification is accompanied by three xml schema files, namely: (TO BE PROVIDED)

815 EDITOR NOTE: The xml schema will be provided with the final version in separated files.

816 5.2 The QualifyingProperties element

817 The QualifyingProperties element acts as a container element for all the qualifying information that is added to
818 an XML signature. The element has the following structure.

```
819 <xsd:element name="QualifyingProperties" type="QualifyingPropertiesType"/>
820 <xsd:complexType name="QualifyingPropertiesType">
821   <xsd:sequence>
822     <xsd:element name="SignedProperties" type="SignedPropertiesType"
823       minOccurs="0"/>
824     <xsd:element name="UnsignedProperties"
825       type="UnsignedPropertiesType"
826       minOccurs="0"/>
827   </xsd:sequence>
828   <xsd:attribute name="Target" type="xsd:anyURI" use="required"/>
829   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
830 </xsd:complexType>
```

831
832 The qualifying properties are split into properties that are cryptographically bound to (i.e. signed by) the XML signature
833 (SignedProperties), and properties that are not cryptographically bound to the XML signature
834 (UnsignedProperties). The SignedProperties shall be covered by a ds:Reference element of the XML
835 signature.

836 The mandatory Target attribute shall refer to the Id attribute of the corresponding ds:Signature. Its value shall
837 be an URI with a bare-name XPointer fragment. When this element is enveloped by the XAdES signature, its
838 not-fragment part shall be empty. Otherwise, its not-fragment part may not be empty.

839 The optional Id attribute can be used to make a reference to the QualifyingProperties container.

840 It is strongly recommended not to include empty xades:SignedProperties or empty
841 xades:UnsignedProperties elements within the signature. Applications verifying XAdES signatures shall
842 ignore empty xades:SignedProperties and empty xades:UnsignedProperties elements.

843 5.2.1 The SignedProperties element

844 The SignedProperties element contains a number of properties that are collectively signed by the XML signature.

845 Below follows the schema definition for SignedProperties element.

```
846 <xsd:element name="SignedProperties" type="SignedPropertiesType" />
847
848 <xsd:complexType name="SignedPropertiesType">
849   <xsd:sequence>
850     <xsd:element name="SignedSignatureProperties"
851       type="SignedSignaturePropertiesType" minOccurs="0"/>
852     <xsd:element name="SignedDataObjectProperties"
853       type="SignedDataObjectPropertiesType" minOccurs="0"/>
854   </xsd:sequence>
855   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
856 </xsd:complexType>
857
```

858 The SignedProperties element may contain properties that qualify the XML signature itself or the signer. If
859 present, they are included as content of the SignedSignatureProperties element, specified in clause 5.2.3.

860 NOTE: If the ds:KeyInfo element is built according to what is specified in clause 4.1.1, it could happen that
861 no signed signature property is required, and no SignedSignatureProperties element would be
862 needed in the XAdES signature.

863 The SignedProperties element may also contain properties that qualify some of the signed data objects. These
864 properties are included as content of the SignedDataObjectProperties element, specified in clause 5.2.4.

865 The optional Id attribute can be used to make a reference to the SignedProperties element.

866 5.2.2 The UnsignedProperties element

867 The UnsignedProperties element contains a number of properties that are not signed by the XML signature.
868 Below follows the schema definition for this element.

```
869 <xsd:element name="UnsignedProperties" type="UnsignedPropertiesType" />
870
871 <xsd:complexType name="UnsignedPropertiesType">
872   <xsd:sequence>
873     <xsd:element name="UnsignedSignatureProperties"
874       type="UnsignedSignaturePropertiesType" minOccurs="0"/>
875     <xsd:element name="UnsignedDataObjectProperties"
876       type="UnsignedDataObjectPropertiesType" minOccurs="0"/>
877   </xsd:sequence>
878   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
879 </xsd:complexType>
880
```

881 The UnsignedProperties element may contain properties that qualify the XML signature itself or the signer.
882 These properties are included as content of the UnsignedSignatureProperties element (see clause 5.2.5).

883 The UnsignedProperties element may also contain properties that qualify some of the signed data objects. These
884 properties are included as content of the UnsignedDataObjectProperties element (see clause 5.2.6).

885 The optional Id attribute can be used to make a reference to the UnsignedProperties element.

886 5.2.3 The SignedSignatureProperties element

887 This element contains properties that qualify the XML signature that has been specified with the Target attribute of
888 the QualifyingProperties container element.

```
889 <xsd:element name="SignedSignatureProperties"
890   type="SignedSignaturePropertiesType" />
891
892 <xsd:complexType name="SignedSignaturePropertiesType">
893   <xsd:sequence>
894     <xsd:element name="SigningTime" type="xsd:dateTime"
895       minOccurs="0"/>
896     <xsd:element name="SigningCertificate" type="CertIDListType"
```

```

897     minOccurs="0"/>
898     <xsd:element name="SignaturePolicyIdentifier"
899       type="SignaturePolicyIdentifierType" minOccurs="0"/>
900     <xsd:element name="SignatureProductionPlace"
901       type="SignatureProductionPlaceType"
902       minOccurs="0"/>
903     <xsd:element name="SignerRole" type="SignerRoleType"
904       minOccurs="0"/>
905     <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
906   </xsd:sequence>
907   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
908 </xsd:complexType>
909

```

910 The optional `Id` attribute can be used to make a reference to the `SignedSignatureProperties` element.

911 The `xsd:any` element shall be used only for ensuring compatibility among different versions of XAdES. By using this element, applications may add any signed signature property defined in any other version of XAdES. It shall not contain elements whose types and contents are defined outside of a XAdES specification.

912 The qualifying property `SigningTime` is described in detail in clause 6.2.1, `SigningCertificate` in clause 6.2.2.1, `SignaturePolicyIdentifier` in clause 6.2.9, `SignatureProductionPlace` in clause 6.2.5, `SignerRole` in clause 6.2.6.1. Finally, the present specification defines `xadesenv111:SigningCertificate` in clause 6.2.2.2, as an alternative to `SigningCertificate`, and `xadesenv111:SignerRole` in clause 6.2.7, as an alternative to `SignerRole`. These two last qualifying properties are defined within the namespace whose URI is `http://uri.etsi.org/19132/v1.1.1#`.

920 5.2.4 The SignedDataObjectProperties element

921 This element contains properties that qualify some of the signed data objects.

```

922 <xsd:element name="SignedDataObjectProperties"
923   type="SignedDataObjectPropertiesType"/>
924
925 <xsd:complexType name="SignedDataObjectPropertiesType">
926   <xsd:sequence>
927     <xsd:element name="DataObjectFormat" type="DataObjectFormatType"
928       minOccurs="0" maxOccurs="unbounded"/>
929     <xsd:element name="CommitmentTypeIndication"
930       type="CommitmentTypeIndicationType" minOccurs="0"
931       maxOccurs="unbounded"/>
932     <xsd:element name="AllDataObjectsTimeStamp" type="XAdESTimeStampType"
933       minOccurs="0" maxOccurs="unbounded"/>
934     <xsd:element name="IndividualDataObjectsTimeStamp"
935       type="XAdESTimeStampType"
936       minOccurs="0" maxOccurs="unbounded"/>
937   </xsd:sequence>
938   <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
939
940   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
941 </xsd:complexType>
942

```

943 The optional `Id` attribute can be used to make a reference to the `SignedDataObjectProperties` element.

944 The `xsd:any` element shall be used only for ensuring compatibility among different versions of XAdES. By using this element, applications may add any signed data object property defined in any other version of XAdES. It shall not contain elements whose types and contents are defined outside of a XAdES specification.

945 The qualifying property `AllDataObjectsTimeStamp` is described in detail in clause 6.2.8.1, `IndividualDataObjectsTimeStamp` in clause 6.2.8.2, `DataObjectFormat` in clause 6.2.4, and `CommitmentTypeIndication` in clause 6.2.3.

950 5.2.5 The UnsignedSignatureProperties element

951 This element contains properties that qualify the XML signature that has been specified with the `Target` attribute of the `QualifyingProperties` container element. The content of this element is not covered by the XML signature.

```

953 <xsd:element name="UnsignedSignatureProperties"
954   type="UnsignedSignaturePropertiesType"/>
955

```

```

956 <xsd:complexType name="UnsignedSignaturePropertiesType">
957   <xsd:choice maxOccurs="unbounded">
958     <xsd:element name="CounterSignature" type="CounterSignatureType" />
959     <xsd:element name="SignatureTimeStamp" type="XAdESTimeStampType"/>
960     <xsd:element name="CompleteCertificateRefs"
961       type="CompleteCertificateRefsType"/>
962     <xsd:element name="CompleteRevocationRefs"
963       type="CompleteRevocationRefsType"/>
964     <xsd:element name="AttributeCertificateRefs"
965       type="CompleteCertificateRefsType"/>
966     <xsd:element name="AttributeRevocationRefs"
967       type="CompleteRevocationRefsType"/>
968     <xsd:element name="SigAndRefsTimeStamp" type="XAdESTimeStampType"/>
969     <xsd:element name="RefsOnlyTimeStamp" type="XAdESTimeStampType"/>
970     <xsd:element name="CertificateValues" type="CertificateValuesType"/>
971     <xsd:element name="RevocationValues" type="RevocationValuesType"/>
972     <xsd:element name="AttrAuthoritiesCertValues"
973       type="CertificateValuesType"/>
974     <xsd:element name="AttributeRevocationValues"
975       type="RevocationValuesType"/>
976     <xsd:element name="ArchiveTimeStamp" type="XAdESTimeStampType"/>
977     <xsd:any namespace="##other" />
978   </xsd:choice>
979   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
980 </xsd:complexType>
981

```

982 The optional `Id` attribute can be used to make a reference to the `UnsignedSignatureProperties` element.

983 The `xsd:any` element shall be used only for ensuring compatibility among different versions of XAdES. By using this
984 element, applications MAY add any unsigned signature property defined in any other version of XAdES. It shall not
985 contain elements whose types and contents are defined outside of a XAdES specification..

986 Some of these qualifying properties are defined within the normative body of the present document. These properties
987 are: `CounterSignature`, which is defined in clause 6.2.7.2, `SignatureTimeStamp` in clause 6.3,
988 `CertificateValues` in clause 6.4.1, `RevocationValues` in clause 6.4.2, `AttrAuthoritiesCertValues`
989 in clause 6.4.3, and `AttributeRevocationValues` in clause 6.4.5.

990 Some of the qualifying properties are defined within the normative Annex A. These properties are:
991 `CompleteCertificateRefs`, which is defined in clause A.1.1.1 of the aforementioned Annex,
992 `CompleteRevocationRefs` in clause A.1.2, `AttributeCertificateRefs` in clause A1.3.1,
993 `AttributeRevocationRefs` in clause A1.4, `SigAndRefsTimeStamp` in clause A.1.5.1,
994 `RefsOnlyTimeStamp` in clause A.1.5.2.

995 Finally, the present document also defines unsigned properties not included in the namespace whose URI is
996 <http://uri.etsi.org/01903/v1.3.2#>. Two of them are defined within the XML namespace whose URI is
997 <http://uri.etsi.org/01903/v1.4.1#>, namely: `xadesv141:ArchiveTimeStamp` specified in clause 6.5.1, and
998 `xadesv141:TimeStampValidationData` specified in clause 6.6. Four of them are defined within the XML
999 XML namespace whose URI is <http://uri.etsi.org/19132/v1.1.1#>, namely:
1000 `xadesenv111:SignaturePolicyStore` in clause 6.2.10, `xadesenv111:RenewedDigests` in clause 6.5.3,
1001 `xadesenv111:CompleteCertificateRefs` in clause A1.1.1.2, and
1002 `xadesenv111:AttributeCertificateRefs` in clause A1.1.3.2.

1003 5.2.6 The `UnsignedDataObjectProperties` element

1004 This element contains properties that qualify some of the signed data objects. The signature generated by the signer
1005 does not cover the content of this element.

```

1006 <xsd:element name="UnsignedDataObjectProperties"
1007   type="UnsignedDataObjectPropertiesType" />
1008
1009 <xsd:complexType name="UnsignedDataObjectPropertiesType">
1010   <xsd:sequence>
1011     <xsd:element name="UnsignedDataObjectProperty" type="AnyType"
1012       maxOccurs="unbounded"/>
1013   </xsd:sequence>
1014   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
1015 </xsd:complexType>
1016

```

1017 The optional `Id` attribute can be used to make a reference to the `UnsignedDataObjectProperties` element.

018 The present document does not specify the usage of any unsigned property qualifying the signed data object. It,
019 however, defines this element for the sake of completeness and to cope with potential future needs for inclusion of such
020 kind of properties. The schema definition leaves open the definition of the contents of this type. The type AnyType is
021 defined in clause 6.1.1.

022 5.3 Incorporating qualifying properties into an XML signature

023 The present document utilizes the `ds:Object` auxiliary element from XMLDSIG [2]. It shall be used to incorporate
024 the qualifying properties into the XMLDSIG signature. In principle, two different means are provided for this
025 incorporation:

- 026 • direct incorporation means that a `QualifyingProperties` element is put as a child of the `ds:Object`;
- 027 • indirect incorporation means that one or more `QualifyingPropertiesReference` elements appear as
028 children of the `ds:Object`. Each one contains information about one `QualifyingProperties` element
029 that is stored in a place different from the signature (see clause 6.3.2).

030 However, the following restrictions apply for using `ds:Object`, `QualifyingProperties` and
031 `QualifyingPropertiesReference`:

- 032 • all instances of the `QualifyingProperties` and the `QualifyingPropertiesReference` elements
033 shall occur within a single `ds:Object` element;
- 034 • at most one instance of the `QualifyingProperties` element may occur within this `ds:Object`
035 element;
- 036 • all signed properties shall occur within a single `QualifyingProperties` element. This element can either
037 be a child of this `ds:Object` element (direct incorporation), or it can be referenced by a
038 `QualifyingPropertiesReference` element. See clause 5.3.1 for information how to sign properties;
- 039 • zero or more instances of the `QualifyingPropertiesReference` element may occur within this
040 `ds:Object` element.

041 No restrictions apply to the relative position of the `ds:Object` containing the `QualifyingProperties` or
042 `QualifyingPropertiesReference` with respect to others `ds:Object` elements present within
043 `ds:Signature`.

044 It is out of the scope of the present document to specify the mechanisms required to guarantee the correct storage of the
045 distributed `QualifyingProperties` elements (i.e. that the properties are stored by the entity that has to store them
046 and that they are not undetectably modified)

047 5.3.1 Signing properties

048 As has already been stated, all the properties that should be protected by the signature have to be collected in a single
049 instance of the `QualifyingProperties` element. Actually these properties are children of the
050 `SignedProperties` child of this element.

051 In order to protect the properties with the signature, a `ds:Reference` element shall be added to the XML signature.
052 This `ds:Reference` element shall be composed in such a way that it uses the `SignedProperties` element
053 mentioned above as the input for computing its corresponding digest.

054 Additionally, applications claiming conformance with the present document shall include the `Type` attribute to this
055 particular `ds:Reference` element, with its value set to:

- 056 • <http://uri.etsi.org/01903#SignedProperties>.

057 This value indicates that the data used for hash computation is a `SignedProperties` element and therefore helps a
058 verifying application to detect the signed properties of a signature conforming to the present document.

059 5.3.2 The QualifyingPropertiesReference element

060 This element contains information about a QualifyingProperties element that is stored in a place different from
061 the signature, for instance in another XML document.

```
062 <xsd:element name="QualifyingPropertiesReference"
063   type="QualifyingPropertiesReferenceType"/>
064
065 <xsd:complexType name="QualifyingPropertiesReferenceType">
066   <xsd:attribute name="URI" type="xsd:anyURI" use="required"/>
067   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
068 </xsd:complexType>
069
```

070 The mandatory URI attribute contains a bare-name XPointer fragment and references an external
071 QualifyingProperties element. Its not-fragment part identifies the enclosing document and its bare-name
072 XPointer fragment identifies the aforementioned element.

073 The optional Id attribute can be used to make a reference to the QualifyingPropertiesReference element.

074 5.4 Managing canonicalization of XML nodesets

075 A number of qualifying properties specified in the present document incorporate optional means for identifying a
076 canonicalization algorithm for computing the canonical form of a certain XML node set.

077 When dealing with legacy XAdES signatures, i.e., signatures created before the publication of the present document,
078 applications claiming conformance to the present document shall assume that absence of the canonicalization indication
079 means that the actual canonicalization algorithm is Canonical XML 1.0 without comments, identified by the URI:

080 <http://www.w3.org/TR/2001/REC-xml-c14n-20010315>

081 When generating new XAdES signatures, applications claiming conformance to the present document shall explicitly
082 indicate the canonicalization algorithm in all the XAdES qualifying properties by including a container for
083 canonicalization algorithm identifier.

084 When upgrading a legacy XAdES signature to an upper form by the generation and incorporation of a certain XAdES
085 qualifying property whose XML Schema definition includes an optional identifier of a canonicalization algorithm,
086 applications claiming conformance to the present document shall include the aforementioned canonicalization algorithm
087 identifier within the qualifying property.

088 Readers are warned that Canonical XML 1.0 does not properly process the inheritance of attributes in the XML
089 namespace (xml:id and xml:base) when canonicalizing document sub-trees. Canonical XML version 1.1 (whose
090 version omitting comments is identified by the URI <http://www.w3.org/2006/12/xml-c14n11>), specifies a variant of the
091 former canonicalization algorithm that properly addresses these issues.

092 Readers are also warned that Canonical XML 1.0 when applied to an XML sub-tree, includes the subtree's ancestor
093 context including all of the namespace declarations and attributes in the "xml:" namespace. The exclusive XML
094 Canonicalization algorithm (whose version omitting comments is identified by the URI
095 <http://www.w3.org/2001/10/xml-exc-c14n#>) completely excludes this ancestor context from the canonicalized sub-tree.

096

097 6 Qualifying properties syntax

098 This clause specifies some of the qualifying properties defined in the present document, as well as a number of auxiliary
099 types.

100 Clause 6.1 defines the auxiliary types.

101 Clause 6.2 specifies the qualifying properties that may appear in XAdES-BES and XAdES-EPES electronic signatures
102 forms as described in clause 4.

103 Clause 6.3 specifies the `SignatureTimeStamp` qualifying properties for XAdES-T electronic signatures forms as
104 described in clause 4.

105 Clause 6.4 specifies the qualifying properties that can contain validation data values.

106 Clause 6.5 specifies qualifying properties for XAdES-A electronic signatures forms as described in clause 4.

107 6.1 Auxiliary syntax

108 The next sub-clauses below specify certain auxiliary XML structures, utilized in several cases throughout the present
109 document.

110 6.1.1 The `AnyType` data type

111 The `AnyType` Schema data type has a content model that allows a sequence of arbitrary XML elements that (mixed
112 with text) is of unrestricted length. It also allows for text content only. Additionally, an element of this data type can
113 bear an unrestricted number of arbitrary attributes. It is used throughout the remaining parts of the present
114 document wherever the content of an XML element has been left open.

```
115 <xsd:complexType name="AnyType" mixed="true">
116   <xsd:sequence minOccurs="0" maxOccurs="unbounded">
117     <xsd:any namespace="##any" processContents="lax"/>
118   </xsd:sequence>
119   <xsd:anyAttribute namespace="##any"/>
120 </xsd:complexType>
```

121 6.1.2 The `ObjectIdentifierType` data type

122 The `ObjectIdentifierType` data type can be used to identify a particular data object.

123 It allows the specification of a unique and permanent identifier of an object. In addition, it may also contain, a textual
124 description of the nature of the data object, and a number of references to documents where additional information
125 about the nature of the data object can be found.

```
126 <xsd:complexType name="ObjectIdentifierType">
127   <xsd:sequence>
128     <xsd:element name="Identifier" type="IdentifierType"/>
129     <xsd:element name="Description" type="xsd:string" minOccurs="0"/>
130     <xsd:element name="DocumentationReferences"
131       type="DocumentationReferencesType" minOccurs="0"/>
132   </xsd:sequence>
133 </xsd:complexType>
```

134 The `Identifier` element contains a permanent identifier. Once the identifier is assigned, it can never be
135 re-assigned again. It supports both the mechanism that is used to identify objects in ASN.1 and the mechanism that is
136 usually used to identify objects in an XML environment:
137

- 138 • in an XML environment objects are typically identified by means of a Uniform Resource Identifier, URI. In
139 this case, the content of `Identifier` consists of the identifying URI, and the optional `Qualifier` attribute
140 does not appear;
- 141 • in ASN.1 an Object Identifier (OID) is used to identify an object. To support an OID, the content of `Identifier`
142 consists of an OID, either encoded as Uniform Resource Name (URN) or as Uniform Resource Identifier
143 (URI). The optional `Qualifier` attribute can be used to provide a hint about the applied encoding (values
144 “OIDAsURN” or “OIDAsURI”).

145 Applications claiming conformance to the present document and needing to identify an object by an OID, should
146 encode it as an URN as specified by the RFC 3061 [10], and set `Qualifier` attribute to “OIDAsURN” value.

- 147 • Should an OID and an URI exist identifying the same object, the present document encourages the use of the
148 URI as explained in the first bullet above.

```

149 <xsd:complexType name="IdentifierType">
150   <xsd:simpleContent>
151     <xsd:extension base="xsd:anyURI">
152       <xsd:attribute name="Qualifier" type="QualifierType"
153         use="optional"/>
154     </xsd:extension>
155   </xsd:simpleContent>
156 </xsd:complexType>
157 <xsd:simpleType name="QualifierType">
158   <xsd:restriction base="xsd:string">
159     <xsd:enumeration value="OIDASURI"/>
160     <xsd:enumeration value="OIDASURN"/>
161   </xsd:restriction>
162 </xsd:simpleType>
163

```

164 The optional `Description` element contains an informal text describing the object identifier.

165 The optional `DocumentationReferences` element consists of an arbitrary number of references pointing to
 166 further explanatory documentation of the object identifier.

```

167 <xsd:complexType name="DocumentationReferencesType">
168   <xsd:sequence maxOccurs="unbounded">
169     <xsd:element name="DocumentationReference" type="xsd:anyURI"/>
170   </xsd:sequence>
171 </xsd:complexType>
172

```

173 6.1.3 The EncapsulatedPKIDataType data type

174 The `EncapsulatedPKIDataType` is used to incorporate non XML pieces of PKI data into an XML structure.
 175 Examples of such PKI data that are widely used at the time being include X.509 certificates and revocation lists, OCSP
 176 responses, attribute certificates and time-stamp tokens.

```

177 <xsd:complexType name="EncapsulatedPKIDataType">
178   <xsd:simpleContent>
179     <xsd:extension base="xsd:base-64Binary">
180       <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
181       <xsd:attribute name="Encoding" type="xsd:anyURI"
182         use="optional"/>
183     </xsd:extension>
184   </xsd:simpleContent>
185 </xsd:complexType>
186

```

187 The content of this data type is the piece of PKI data, base-64 encoded as defined in [2].

188 The `Encoding` attribute is an URI identifying the encoding used in the original PKI data. So far, the following URIs
 189 have been identified:

- 190 • <http://uri.etsi.org/01903/v1.2.2#DER> for denoting that the original PKI data were ASN.1 data encoded in
 191 DER.
- 192 • <http://uri.etsi.org/01903/v1.2.2#BER> for denoting that the original PKI data were ASN.1 data encoded in BER.
- 193 • <http://uri.etsi.org/01903/v1.2.2#CER> for denoting that the original PKI data were ASN.1 data encoded in CER.
- 194 • <http://uri.etsi.org/01903/v1.2.2#PER> for denoting that the original PKI data were ASN.1 data encoded in PER.
- 195 • <http://uri.etsi.org/01903/v1.2.2#XER> for denoting that the original PKI data were ASN.1 data encoded in
 196 XER.

197 If the `Encoding` attribute is not present, then it is assumed that the PKI data is ASN.1 data encoded in DER.

198 NOTE: In some clauses of the present document, specific XAdES properties related to these data restrict the
 199 encoding options to only one certain type of the aforementioned PKI data.

200 The optional `ID` attribute can be used to make a reference to an element of this data type

201 6.1.4 Types for time-stamp tokens management

202 XAdES uses time-stamp tokens in a number of use cases. The present document defines:

- 203 • A XML schema definition of an abstract base type and two concrete derived types used as containers for
204 time-stamp tokens.
- 205 • A number of properties of one of the aforementioned concrete types. Time-stamp tokens included in these
206 properties will cover a specific set of elements and properties of XAdES signature forms and will satisfy, in
207 this way, different requirements.

208 A time-stamp token is obtained by sending the digest value of the given data (message imprint henceforth) to the
209 Time-Stamp Authority (TSA). The returned time-stamp token is a signed data that contains the digest value, the identity
210 of the TSA, and the time of stamping. This proves that the given data existed *before* the time of stamping.

211 NOTE: readers should note that the term time-stamp token used throughout the present document does NOT refer
212 to the TSA's response to a requesting client, but the token generated by the TSA, which is present within
213 this response. In the case of RFC 3161 [8] protocol, the time-stamp token term is referring to the
214 `timestampToken` field within the `TimestampResp` element (the TSA's response returned to the
215 requesting client).

216 XAdES time-stamp tokens container properties contain time-stamp tokens computed on both, elements defined in
217 XMLDSIG [2] and properties defined in the present document. The present document uses the term time-stamped data
218 objects for indistinctly denoting any of them.

219 6.1.4.1 Time-stamp properties in XAdES

220 Below follows the list of the properties containing time-stamps that are defined by the present document:

- 221 • Properties that contain time-stamp tokens proving that some or all the data objects to be signed have been
222 created before some time: `AllDataObjectsTimeStamp` and
223 `IndividualDataObjectsTimeStamp`.
- 224 • `SignatureTimeStamp`: it is a container for a time-stamp token over the `SignatureValue` element to
225 protect against repudiation in case of a key compromise.
- 226 • To provide for long term validity of an XML signature, the signature and validation data values are
227 time-stamped. `xadesv141:ArchiveTimeStamp` and `xadesenv111:ArchiveTimeStamp` are
228 defined for this purpose. More than one instance of these properties can be added as time goes on to the
229 archived electronic signature
- 230 • Annex A specifies two properties that contain time-stamp tokens on properties defined in that annex.

231 6.1.4.2 The `GenericTimeStampType` data type

232 The abstract base container type for time-stamp tokens specified by the present document does have the following
233 features:

- 234 • It may contain encapsulated RFC 3161 [8] time-stamp tokens as well as XML time-stamp tokens.
- 235 • It may contain more than one time-stamp token generated for the same XAdES data objects (each one issued
236 by different TSAs, for instance).
- 237 • It provides means for managing time-stamp tokens computed on XAdES data objects (for instance XAdES
238 properties) or time-stamp tokens computed on external data.
- 239 • It may use specific elements for identifying what is time-stamped and how to generate the input data for the
240 computation of the digest value to be sent to the TSA. For certain XAdES data objects under certain
241 circumstances this information may be implicit.

242 Below follows the schema definition for the data type.

243 `<xsd:element name="Include" type="IncludeType"/>`

```

244 <xsd:complexType name="IncludeType">
245   <xsd:attribute name="URI" type="xsd:anyURI" use="required"/>
246   <xsd:attribute name="referencedData" type="xsd:boolean"
247     use="optional"/>
248 </xsd:complexType>
249
250 <xsd:element name="ReferenceInfo" type="ReferenceInfoType"/>
251 <xsd:complexType name="ReferenceInfoType">
252   <xsd:sequence>
253     <xsd:element ref="ds:DigestMethod"/>
254     <xsd:element ref="ds:DigestValue"/>
255   </xsd:sequence>
256   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
257   <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
258 </xsd:complexType>
259
260 <xsd:complexType name="GenericTimeStampType" abstract="true">
261   <xsd:sequence>
262     <xsd:choice minOccurs="0">
263       <xsd:element ref="Include" minOccurs="0" maxOccurs="unbounded"/>
264       <xsd:element ref="ReferenceInfo" maxOccurs="unbounded"/>
265     </xsd:choice>
266     <xsd:element ref="ds:CanonicalizationMethod" minOccurs="0"/>
267     <xsd:choice maxOccurs="unbounded">
268       <xsd:element name="EncapsulatedTimeStamp"
269         type="EncapsulatedPKIDataType"/>
270       <xsd:element name="XMLTimeStamp" type="AnyType"/>
271     </xsd:choice>
272   </xsd:sequence>
273   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
274 </xsd:complexType>
275

```

The optional `ds:CanonicalizationMethod` element indicates the canonicalization algorithm used for canonicalizing XML node sets resulting after retrieving (and processing when required) the data objects covered by the time-stamp token(s). Clause 5.4 of the present document specifies the requirements that applications claiming conformance to the present document shall satisfy when dealing with this element.

The time-stamp token generated by the TSA can be either an ASN.1 data object (as defined in [8], use `EncapsulatedTimeStamp`), or it can be encoded as XML (use `XMLTimeStamp`).

Details on the different elements and supporting types are given in the clauses that define the two concrete types: `XAdESTimeStampType` and `OtherTimeStampType`.

6.1.4.3 The `XAdESTimeStampType` data type

This concrete derived type is provided for containing time-stamp tokens computed on data objects of XAdES signatures. Applications claiming alignment with the present document shall implement it because all the properties listed in clause 6.1.4.1 are elements of this type.

Below follows the schema definition for the data type.

```

289 <xsd:element name="XAdESTimeStamp" type="XAdESTimeStampType"/>
290
291 <xsd:complexType name="XAdESTimeStampType">
292   <xsd:complexContent>
293     <xsd:restriction base="GenericTimeStampType">
294       <xsd:sequence>
295         <xsd:element ref="Include" minOccurs="0"
296           maxOccurs="unbounded"/>
297         <xsd:element ref="ds:CanonicalizationMethod" minOccurs="0"/>
298         <xsd:choice maxOccurs="unbounded">
299           <xsd:element name="EncapsulatedTimeStamp"
300             type="EncapsulatedPKIDataType"/>
301           <xsd:element name="XMLTimeStamp" type="AnyType"/>
302         </xsd:choice>
303       </xsd:sequence>
304       <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
305     </xsd:restriction>
306   </xsd:complexContent>
307 </xsd:complexType>
308

```

This type provides two mechanisms for identifying data objects that are covered by the time-stamp token present in the container, and for specifying how to compute the time-stamp token's message imprint:

- 311 • Explicit. This mechanism uses the `Include` element for referencing specific data objects and for indicating
312 their contribution to the input of the message imprint's computation.
- 313 • Implicit. For certain time-stamp container properties under certain circumstances, applications do not require
314 any additional indication for knowing that certain data objects are covered by the time-stamp tokens and how
315 they contribute to the input of the message imprint's computation. The present document specifies, in the
316 clauses defining such properties (clauses 6.2.8.1, 6.3, 6.5), how applications shall act in these cases without
317 explicit indications.

318 Clause 6.1.4.3.1 shows the principles that govern the explicit indication mechanism.

319 6.1.4.3.1 Include mechanism

320 `Include` elements explicitly identify data objects that are time-stamped. Their order of appearance indicates how the
321 data objects contribute in the generation of the input to the time-stamp's message imprint computation.

322 The `IndividualDataObjectsTimeStamp` time-stamp token container property uses this mechanism. Each
323 `Include` element shall contain an URI referencing one of the `ds:Reference` elements in the XAdES signature.

324 Two unsigned properties specified in Annex A clauses A.1.5.1 and A.1.5.2 also use this mechanism under certain
325 circumstances.

326 The URI attribute in `Include` element identifies one time-stamped data object. Its value shall follow the rules
327 indicated below:

- 328 • It shall have an empty not-fragment part and a bare-name XPointer fragment when the `Include` and the
329 time-stamped data object are in the same document.
- 330 • It shall have a not empty not-fragment part and a bare-name XPointer fragment when the `Include` and the
331 time-stamped data object are not in the same document.
- 332 • When not empty, its not-fragment part shall be equal to:
 - 333 - the not-fragment part of the `Target` attribute of the `QualifyingProperties` enclosing the
334 `Include` element if the time-stamped data object is enveloped by the XAdES signature; or
 - 335 - the not-fragment part of the URI attribute of the `QualifyingPropertiesReference` element
336 referencing the `QualifyingProperties` element enveloping the time-stamped data object if this
337 `QualifyingProperties` element is not enveloped by the XAdES signature.

338 Applications aligned with the present document shall parse the retrieved resource, and then process the bare-name
339 XPointer as explained below to get a XPath node-set suitable for being processed according to the selected
340 canonicalization algorithm. For processing the bare-name XPointer, applications shall use as XPointer evaluation
341 context the root node of the XML document that contains the element referenced by the not-fragment part of URI.
342 Applications shall derive an XPath node-set from the resultant location-set as indicated below:

- 343 1) Replace the element node E retrieved by the bare-name XPointer with E plus all descendants of E (text,
344 comments, PIs, elements) and all namespace and attribute nodes of E and its descendant elements.
- 345 2) Delete all the comment nodes.

346 In time-stamps that cover `ds:Reference` elements, the attribute `referencedData` may be present. If present with
347 value set to `"true"`, the time-stamp shall be computed on the result of processing the corresponding `ds:Reference`
348 element according to the XMLDSIG processing model. If the attribute is not present or is present with value `"false"`,
349 the time-stamp shall be computed on the `ds:Reference` element itself. When appearing in a time-stamp container
350 property, each `Include` element shall be processed in order as detailed below:

- 351 1) Retrieve the data object referenced in the URI attribute following the referencing mechanism indicated above.
- 352 2) If the retrieved data is a `ds:Reference` element and the `referencedData` attribute is set to the value
353 `"true"`, take the result of processing the retrieved `ds:Reference` element according to the reference
354 processing model of XMLDSIG; otherwise take the `ds:Reference` element itself.

- 355 3) If the resulting data is an XML node set, canonicalize it as specified in clause 5.4.
- 356 4) Concatenate the resulting octets to those resulting from previous processing as indicated in the corresponding
357 time-stamp container property.

358 6.1.4.4 The OtherTimeStampType data type

359 This concrete derived type is provided for containing time-stamp tokens computed on a collection of data objects that
360 are not present in the XAdES signature.

361 Below follows the schema definition for the data type.

```
362 <xsd:element name="OtherTimeStamp" type="OtherTimeStampType"/>
363
364 <xsd:complexType name="OtherTimeStampType">
365   <xsd:complexContent>
366     <xsd:restriction base="GenericTimeStampType">
367       <xsd:sequence>
368         <xsd:element ref="ReferenceInfo" maxOccurs="unbounded"/>
369         <xsd:element ref="ds:CanonicalizationMethod" minOccurs="0"/>
370         <xsd:choice>
371           <xsd:element name="EncapsulatedTimeStamp"
372             type="EncapsulatedPKIDataType"/>
373           <xsd:element name="XMLTimeStamp" type="AnyType"/>
374         </xsd:choice>
375       </xsd:sequence>
376       <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
377     </xsd:restriction>
378   </xsd:complexContent>
379 </xsd:complexType>
380
```

381 Each ReferenceInfo element contains the digest of one external data object. Attribute URI identifies the data
382 object. As in XMLDSIG, if it is omitted, the application is expected to know the identity of the referenced object.
383 Attribute Id permits this element to be referenced from elsewhere. Element ds:DigestMethod identifies the digest
384 algorithm applied to the external data object. Element ds:DigestValue contains the base-64 encoded value of the
385 digest of the referenced data object.

386 Attribute Id and elements ds:CanonicalizationMethod, EncapsulatedTimeStamp and
387 XMLTimeStamp will be used exactly as in XAdESTimeStampType.

388 For this type the actual input to the computation of the message imprint that will be sent to the TSA is the concatenation
389 of the present ReferenceInfo elements, canonicalized as specified in clause 5.4.

390 The implementation of such a type is not mandatory for applications that claim conformance to the present document,
391 as it does not define any property of this type

392 6.2 Properties for XAdES-BES and XAdES-EPES forms

393 This clause describes in detail the qualifying properties that can appear in XAdES-BES and XAdES-EPES forms as
394 described in clauses 4.1.1 and 4.1.2.

395 6.2.1 The SigningTime element

396 The SigningTime property is an optional signed property that qualifies the whole signature. There shall be at most
397 one occurrence of this property in the signature.

398 The SigningTime property specifies the time at which the signer (purportedly) performed the signing process.

399 Below follows the Schema definition for this element.

```
400 <xsd:element name="SigningTime" type="xsd:dateTime"/>
401
```

402 6.2.2 References to the signing certificate

403 Sub-clauses below specify two signed properties qualifying the signature used as containers of a reference and the
 404 digest value of the signing certificate (and optionally to other certificates in its certification path), namely:
 405 `SigningCertificate` and `xadesenv111:SigningCertificate`.

406 Only one of these two properties may be incorporated to a XAdES signature: if one of these properties is incorporated,
 407 then the other one shall not be incorporated.

408 **EDITOR NOTE:** feedback from stakeholders is requested on the solution proposed: define the new
 409 `xadesenv111:SigningCertificate` for acknowledging deprecation of `ds:X509IssuerSerial`, but keep
 410 `xades:SigningCertificate` for keeping backwards compatibility.

411 6.2.2.1 The `SigningCertificate` element

412 The `SigningCertificate` property is a signed property that qualifies the signature. At most one
 413 `SigningCertificate` element may be present in the signature.

414 The `SigningCertificate` property contains references to certificates and digest values computed on their DER-
 415 encodings.

416 Below follows the Schema definition.

```

417 <xsd:element name="SigningCertificate" type="CertIDListType"/>
418
419 <xsd:complexType name="CertIDListType">
420   <xsd:sequence>
421     <xsd:element name="Cert" type="CertIDType"
422       maxOccurs="unbounded"/>
423   </xsd:sequence>
424 </xsd:complexType>
425
426 <xsd:complexType name="CertIDType">
427   <xsd:sequence>
428     <xsd:element name="CertDigest" type="DigestAlgAndValueType"/>
429     <xsd:element name="IssuerSerial" type="ds:X509IssuerSerialType"/>
430   </xsd:sequence>
431   <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
432 </xsd:complexType>
433
434 <xsd:complexType name="DigestAlgAndValueType">
435   <xsd:sequence>
436     <xsd:element ref="ds:DigestMethod"/>
437     <xsd:element ref="ds:DigestValue"/>
438   </xsd:sequence>
439 </xsd:complexType>
440
```

441 The `SigningCertificate` element contains the aforementioned sequence of certificate identifiers and digests
 442 computed on the certificates (`Cert` elements).

443 The element `IssuerSerial` contains the identifier of the referenced certificate. Should the
 444 `ds:X509IssuerSerial` element appear in the signature to denote the same certificate, its value shall be consistent
 445 with the corresponding `IssuerSerial` element.

446 The element `CertDigest` contains the digest of the referenced certificate. It contains two elements:
 447 `ds:DigestMethod` indicates the digest algorithm and `ds:DigestValue` contains the base-64 encoded value of
 448 the digest computed on the DER-encoded certificate.

449 The optional URI attribute provides an indication of where the referenced certificate may be found. It is intended that
 450 this attribute is be used as a hint, as implementations may have alternative ways for retrieving the referenced certificate
 451 if it is not found at the referenced place.

452 **EDITOR NOTE:** clarification of the usage of this attribute. If the certificate is not found anymore at the place
 453 referenced by the URI, implementations may still be able to retrieve it.

454 The certificate used to verify the signature shall be present in this property. Other certificates may also be present in the
 455 sequence, which may include all the certificates up to the point of trust.

456 6.2.2.2 The `xadesenv111:SigningCertificate` element

457 The `xadesenv111:SigningCertificate` property is a signed property that qualifies the signature. At most one
458 `xadesenv111:SigningCertificate` element may be present in the signature.

459 The `xadesenv111:SigningCertificate` property contains references to certificates and digest values
460 computed on their DER-encodings.

461 Below follows the Schema definition.

```
462 <!-- targetNamespace="http://uri.etsi.org/19132/v1.1.1#" -->
463
464 <xsd:element name="SigningCertificate" type="xadesenv111:CertIDListType"/>
465
466 <xsd:complexType name="CertIDListType">
467   <xsd:sequence>
468     <xsd:element name="Cert" type="xadesenv111:CertIDType" maxOccurs="unbounded"/>
469   </xsd:sequence>
470 </xsd:complexType>
471
472 <xsd:complexType name="CertIDType">
473   <xsd:sequence>
474     <xsd:element name="CertDigest" type="xades:DigestAlgAndValueType"/>
475     <xsd:element name="IssuerSerial" type="xadesenv111:IssuerSerialType"/>
476   </xsd:sequence>
477   <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
478 </xsd:complexType>
479
480 <xsd:complexType name="IssuerSerialType">
481   <xsd:sequence>
482     <xsd:element name="X509IssuerName" type="xsd:string"/>
483     <xsd:element name="X509SerialNumber" type="xsd:string"/>
484   </xsd:sequence>
485 </xsd:complexType>
```

487 The semantics of the elements and types defined above are identical to the elements and types defined in clause 6.2.2.1,
488 with the only exception of the new element `xadesenv111:IssuerSerial` element, of type
489 `xadesenv111:IssuerSerialType`. Child `xadesenv111:X509IssuerName` has the same syntax and
490 semantics than `ds:X509IssuerName`. The content of child `xadesenv111:X509SerialNumber` shall be a
491 string containing the textual representation of the serial number field of the referenced certificate in base 10 without
492 leading zeroes.

493 **NOTE:** The reason for defining the new `xadesenv111:SigningCertificate` signed property is that
494 XMLDSig in its version 1.1 [2], has deprecated the usage of `ds:X509IssuerSerial` element. The
495 rationale for its deprecation is that some XML Schema validation tools do not deal with integer values
496 that have more than 18 decimal digits. It is not uncommon that the randomly generated serial numbers
497 need more than 18 digits.

498 **EDITOR NOTE:** The `ds:X509IssuerSerial` element is a mandatory element of `SigningCertificate`
499 element for keeping semantic alignment with CAAdES and because it provides readable information to
500 human beings on the referenced certificate. These are the reasons why the
501 `xadesenv111:Issuerserial` element is specified as mandatory in the new
502 `xadesenv111:CertIDType`. Feedback is kindly requested to stakeholders on the suitability of
503 defining the new `xadesenv111:SigningCertificate` and on its specification itself.

504

505 6.2.3 The `CommitmentTypeIndication` element

506 The `CommitmentTypeIndication` property is an optional signed property that qualifies signed data object(s). A
507 XAdES signature aligned with the present document may contain more than one `CommitmentTypeIndication`
508 elements.

509 The `CommitmentTypeIndication` property contains an explicit indication of the type of the commitment made by
510 the signatory when signing a certain data object.

511 Below follows the schema definition for this element.

```

512 <xsd:element name="CommitmentTypeIndication" type="CommitmentTypeIndicationType"/>
513
514 <xsd:complexType name="CommitmentTypeIndicationType">
515   <xsd:sequence>
516     <xsd:element name="CommitmentTypeId"
517       type="ObjectIdentifierType"/>
518     <xsd:choice>
519       <xsd:element name="ObjectReference" type="xsd:anyURI"
520         maxOccurs="unbounded"/>
521       <xsd:element name="AllSignedDataObjects"/>
522     </xsd:choice>
523     <xsd:element name="CommitmentTypeQualifiers"
524       type="CommitmentTypeQualifiersListType" minOccurs="0"/>
525   </xsd:sequence>
526 </xsd:complexType>
527
528 <xsd:complexType name="CommitmentTypeQualifiersListType">
529   <xsd:sequence>
530     <xsd:element name="CommitmentTypeQualifier"
531       type="AnyType" minOccurs="0" maxOccurs="unbounded"/>
532   </xsd:sequence>
533 </xsd:complexType>
534

```

535 The `CommitmentTypeId` element univocally identifies the type of commitment made by the signer. Below follows a
536 list of commitment types and their corresponding URIs:

- 537 • **Proof of origin** indicates that the signer recognizes to have created, approved and sent the signed data object.
538 The URI for this commitment is <http://uri.etsi.org/01903/v1.2.2#ProofOfOrigin>.
- 539 • **Proof of receipt** indicates that signer recognizes to have received the content of the signed data object. The
540 URI for this commitment is <http://uri.etsi.org/01903/v1.2.2#ProofOfReceipt>.
- 541 • **Proof of delivery** indicates that the TSP providing that indication has delivered a signed data object in a local
542 store accessible to the recipient of the signed data object. The URI for this commitment is
543 <http://uri.etsi.org/01903/v1.2.2#ProofOfDelivery>.
- 544 • **Proof of sender** indicates that the entity providing that indication has sent the signed data object (but not
545 necessarily created it). The URI for this commitment is <http://uri.etsi.org/01903/v1.2.2#ProofOfSender>.
- 546 • **Proof of approval** indicates that the signer has approved the content of the signed data object. The URI for
547 this commitment is <http://uri.etsi.org/01903/v1.2.2#ProofOfApproval>.
- 548 • **Proof of creation** indicates that the signer has created the signed data object (but not necessarily approved, nor
549 sent it). The URI for this commitment is <http://uri.etsi.org/01903/v1.2.2#ProofOfCreation>

550 One `ObjectReference` element refers to one `ds:Reference` element of the `ds:SignedInfo` or a signed
551 `ds:Manifest`, corresponding with one data object qualified by this property. If some but not all the signed data
552 objects share the same commitment, one `ObjectReference` element shall appear for each one of them. However, if
553 all the signed data objects share the same commitment, the `AllSignedDataObjects` empty element shall be
554 present.

555 The `CommitmentTypeQualifiers` element provides means to include additional qualifying information on the
556 commitment made by the signatory.

557 6.2.4 The `DataObjectFormat` element

558 The `DataObjectFormat` property is an optional signed property that qualifies one specific signed data object. In
559 consequence, a XAdES signature aligned with the present document may contain more than one
560 `DataObjectFormat` elements, each one qualifying one signed data object.

561 The `DataObjectFormat` element provides information that describes the format of the signed data object. This
562 element should be present when the signed data is to be presented to human users on validation and the presentation
563 format is not implicit within the data that has been signed.

564 Below follows the schema definition for this element.

```
565 <xsd:element name="DataObjectFormat" type="DataObjectFormatType"/>
566
567 <xsd:complexType name="DataObjectFormatType">
568   <xsd:sequence>
569     <xsd:element name="Description" type="xsd:string" minOccurs="0"/>
570     <xsd:element name="ObjectIdentifier" type="ObjectIdentifierType"
571       minOccurs="0"/>
572     <xsd:element name="MimeType" type="xsd:string" minOccurs="0"/>
573     <xsd:element name="Encoding" type="xsd:anyURI" minOccurs="0"/>
574   </xsd:sequence>
575   <xsd:attribute name="ObjectReference" type="xsd:anyURI"
576     use="required"/>
577 </xsd:complexType>
578
```

579 The mandatory `xades:ObjectReference` attribute shall reference the `ds:Reference` child of the
580 `ds:SignedInfo` or a signed `ds:Manifest` element referencing the signed data object qualified by this property.

581 This element can convey:

- 582 • textual information related to the signed data object in element `Description`;
- 583 • an identifier indicating the type of the signed data object in element `ObjectIdentifier`;
- 584 • an indication of the MIME type of the signed data object in element `MimeType`;
- 585 • an indication of the encoding format of the signed data object in element `Encoding`.

586 At least one element of `Description`, `ObjectIdentifier` and `MimeType` shall be present within the property.

587 If the `DataObjectFormat` property references a `ds:Reference` that in turn references a `ds:Object` within the
588 XAdES signature, and if this `ds:Object` element has the `MimeType` or (and) the `Encoding` attribute(s), then
589 `DataObjectFormat`'s children `MimeType` and `Encoding` shall have exactly the same values when present.

590 6.2.5 The `SignatureProductionPlace` element

591 The `SignatureProductionPlace` property is an optional signed property that qualifies the signer. There shall be
592 at most one occurrence of this property in the signature.

593 The `SignatureProductionPlace` element specifies an address associated with the signer at a particular
594 geographical (e.g. city) location.

595 Below follows the schema definition for this element.

```
596 <xsd:element name="SignatureProductionPlace" type="SignatureProductionPlaceType"/>
597
598 <xsd:complexType name="SignatureProductionPlaceType">
599   <xsd:sequence>
600     <xsd:element name="City" type="xsd:string" minOccurs="0"/>
601     <xsd:element name="StateOrProvince" type="xsd:string"
602       minOccurs="0"/>
603     <xsd:element name="PostalCode" type="xsd:string" minOccurs="0"/>
604     <xsd:element name="CountryName" type="xsd:string" minOccurs="0"/>
605   </xsd:sequence>
606 </xsd:complexType>
607
```

608 6.2.6 Elements for incorporating signer attributes

609 XAdES signatures allow the incorporation of optional signed properties encapsulating signer attributes (e.g. role). The
610 present document differentiates two types of attributes:

- 611 • attributes claimed by the signer;
- 612 • attributes certified by a trusted authority.

613 Sub-clauses below specify two ways of incorporating signer attributes in a XAdES signature.

6.2.6.1 The SignerRole element

The SignerRole property is an optional unsigned property that qualifies the signer. There shall be at most one occurrence of this property within a XAdES signature.

Below follows the Schema definition for this element:

```

518 <xsd:element name="SignerRole" type="SignerRoleType"/>
519
520 <xsd:complexType name="SignerRoleType">
521   <xsd:sequence>
522     <xsd:element name="ClaimedRoles" type="ClaimedRolesListType"
523       minOccurs="0"/>
524     <xsd:element name="CertifiedRoles" type="CertifiedRolesListType"
525       minOccurs="0"/>
526   </xsd:sequence>
527 </xsd:complexType>
528
529 <xsd:complexType name="ClaimedRolesListType">
530   <xsd:sequence>
531     <xsd:element name="ClaimedRole" type="AnyType"
532       maxOccurs="unbounded"/>
533   </xsd:sequence>
534 </xsd:complexType>
535
536 <xsd:complexType name="CertifiedRolesListType">
537   <xsd:sequence>
538     <xsd:element name="CertifiedRole" type="EncapsulatedPKIDataType"
539       maxOccurs="unbounded"/>
540   </xsd:sequence>
541 </xsd:complexType>
542

```

This property contains a sequence of roles that the signer can play (element SignerRole). At least one of the two elements ClaimedRoles or CertifiedRoles shall be present.

The ClaimedRoles element contains a sequence of roles claimed by the signer but not certified. Additional contents types may be defined on a domain application basis and be part of this element. The namespaces given to the corresponding XML schemas will allow their unambiguous identification in the case these attributes are expressed in XML syntax (e.g. SAML assertions of different versions).

The CertifiedRoles element contains the base-64 encoding of one or more DER-encoded attribute certificates as specified within ITU-T X.509 [5], for the signer.

6.2.6.2 The xadesenv111:SignerRole element

The xadesenv111:SignerRole property is an optional unsigned property that qualifies the signer. There shall be at most one occurrence of this property within a XAdES signature.

Below follows the Schema definition for this element. Note that the elements are defined within the xadesenv111 namespace, whose URI is <http://uri.etsi.org/19132/v1.1.1#>.

```

556 <!-- targetNamespace="http://uri.etsi.org/19132/v1.1.1#" -->
557
558 <xsd:element name="SignerRole" type="xadesenv111:SignerRoleType"/>
559
560 <xsd:complexType name="SignerRoleType">
561   <xsd:sequence>
562     <xsd:element name="ClaimedRoles" type="xades:ClaimedRolesListType" minOccurs="0"/>
563     <xsd:element name="CertifiedRoles" type="xadesenv111:CertifiedRolesListType" minOccurs="0"/>
564     <xsd:element name="SignedAssertions" type="xadesenv111:SignedAssertionsListType"
565       minOccurs="0"/>
566   </xsd:sequence>
567 </xsd:complexType>
568
569 </xsd:complexType>
570 <xsd:complexType name="CertifiedRolesListType">
571   <xsd:sequence>
572     <xsd:element name="CertifiedRole" type="xadesenv111:CertifiedRoleType"
573       maxOccurs="unbounded"/>
574   </xsd:sequence>
575 </xsd:complexType>
576

```

```

577 <xsd:complexType name="SignedAssertionsListType">
578   <xsd:sequence>
579     <xsd:element name="SignedAssertion" type="xades:AnyType" maxOccurs="unbounded"/>
580   </xsd:sequence>
581 </xsd:complexType>
582
583 <xsd:complexType name="CertifiedRoleType">
584   <xsd:choice>
585     <xsd:element name="x509AttributeCertificate" type="xades:EncapsulatedPKIDataType"/>
586     <xsd:element name="otherAttributeCertificate" type="xades:AnyType"/>
587   </xsd:choice>
588 </xsd:complexType>
589

```

590 The `ClaimedRoles` element has the same semantics and syntax as in `xades:SignerRole`.

591 The `CertifiedRoles` element may contain:

- 592 • the base-64 encoding of one or more DER-encoded X509 attribute certificates for the signer within the
593 `x509AttributeCertificate` element or
- 594 • attribute certificates (issued, in consequence, by Attribute Authorities) in different syntax than the one used for
595 X509 attribute certificates, within the `otherAttributeCertificate` element. The definition of specific
596 `otherAttributeCertificates` is outside of the scope of the present document.

597 The `SignedAssertions` element contains signed assertions (e.g. signed SAML assertions). These assertions are
598 signed by entities that do not satisfy all the requirements specified for being considered Attribute Authorities. The
599 definition of specific `signerAssertions` is outside of the scope of the present document.

700 NOTE 1: EN 319 411-4 [i.1], annex A specifies general requirements for attribute certificates

701 6.2.7 Countersignatures

702 This clause defines two standard mechanisms for managing countersignatures. Details are given in clauses below.

703 6.2.7.1 Countersignature identifier in `Type` attribute of `ds:Reference`

704 The present document defines the following URI value:

- 705 • <http://uri.etsi.org/01903#CountersignedSignature>.

706 A XAdES signature containing a `ds:Reference` element whose `Type` attribute has this value will indicate that it is,
707 in fact, a countersignature of the signature referenced by this element. The `ds:Reference` element shall be built so
708 that the countersignature actually signs the `ds:SignatureValue` element of the countersigned signature. All the
709 XMLDSIG rules apply in the processing of the aforementioned `ds:Reference` element. The only purpose of this
710 definition is to serve as an easy identification of a signature as actually being a countersignature.

711 6.2.7.2 Enveloped countersignatures: the `CounterSignature` element

712 The `CounterSignature` property is an optional unsigned property that qualifies the signature. A XAdES signature
713 may have more than one `CounterSignature` properties.

714 The `CounterSignature` property contains one countersignature of the qualified signature.

715 Below follows the schema definition for this element.

```

716 <xsd:element name="CounterSignature" type="CounterSignatureType" />
717
718 <xsd:complexType name="CounterSignatureType">
719   <xsd:sequence>
720     <xsd:element ref="ds:Signature"/>
721   </xsd:sequence>
722 </xsd:complexType>

```

723

724 The content of this property is a XMLDSIG or XAdES signature whose `ds:SignedInfo` shall contain one
 725 `ds:Reference` element referencing the `ds:SignatureValue` element of the embedding and countersigned
 726 XAdES signature.

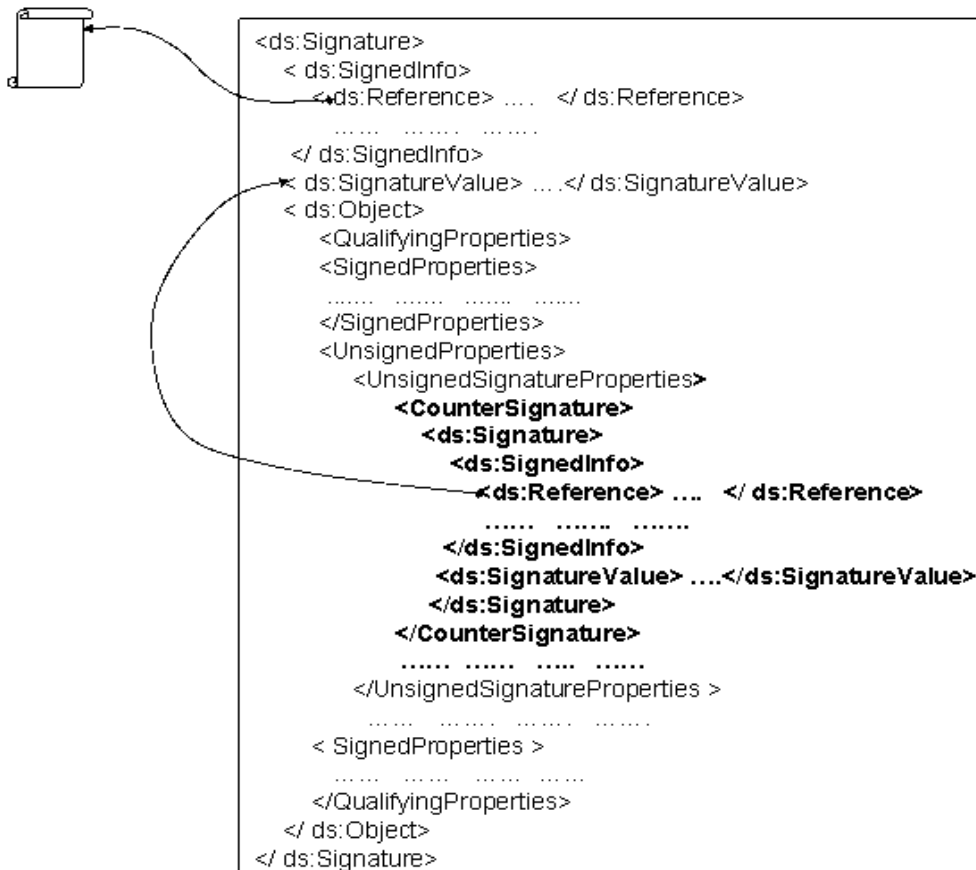
727 The content of the `ds:DigestValue` in the aforementioned `ds:Reference` element of the countersignature shall
 728 be the base-64 encoded digest of the complete (and canonicalized) `ds:SignatureValue` element (i.e. including the
 729 starting and closing tags) of the embedding and countersigned XAdES signature. Applications shall build this
 730 `ds:Reference` accordingly, using any of the mechanisms specified by XMLDSIG for achieving this objective. By
 731 doing this the countersignature actually signs the `ds:SignatureValue` element of the embedding XAdES signature.

732 Applications may add other `ds:Reference` elements referencing the `ds:SignatureValue` elements of
 733 previously existent `CounterSignature` elements. This allows for building arbitrarily long chains of explicit
 734 countersignatures.

735 A countersignature may itself be qualified by a `CounterSignature` property, which will have a `ds:Reference`
 736 element referencing the `ds:SignatureValue` of the first countersignature, built as described above. This is an
 737 alternative way of constructing arbitrarily long series of countersignatures, each one signing the
 738 `ds:SignatureValue` element of the one where it is directly embedded.

739 If the countersignature is a XAdES signature, its production shall follow the rules dictated by the present document.

740 Below follows the schema definition for this element.



741

742

Figure 1: Use of CounterSignature element

743

744 6.2.8 Time-stamps on signed data objects

745 6.2.8.1 The AllDataObjectsTimeStamp element

746 The AllDataObjectsTimeStamp element is an optional signed property. Several instances of this property from
747 different TSAs may occur within one XAdES signature.

748 The AllDataObjectsTimeStamp element contains the time-stamp computed before the signature production, over
749 the sequence formed by ALL the ds:Reference elements within the ds:SignedInfo in their order of appearance
750 referencing, whatever the signer wants to sign except the SignedProperties element.

751 Below follows the schema definition for this element.

```
752 <xsd:element name="AllDataObjectsTimeStamp" type="XAdESTimeStampType"/>
```

753
754 This property uses the Implicit mechanism. The input to the computation of the message imprint shall be the result of
755 processing the aforementioned suitable ds:Reference elements in their order of appearance within
756 ds:SignedInfo as follows:

- 757 1) Process the retrieved ds:Reference element according to the reference processing model of XMLDSIG.
- 758 2) If the result is a XML node set, canonicalize it as specified in clause 5.4.
- 759 3) Concatenate the resulting octets to those resulting from previously processed ds:Reference elements in
760 ds:SignedInfo.

761 6.2.8.2 The IndividualDataObjectsTimeStamp element

762 The IndividualDataObjectsTimeStamp element is an optional signed property that qualifies the signed data
763 object(s). Several instances of this property can occur within the one XAdES signature.

764 The IndividualDataObjectsTimeStamp element contains the time-stamp computed before the signature
765 production, over a sequence formed by SOME ds:Reference elements within the ds:SignedInfo and any
766 signed ds:Manifest. Note that this sequence cannot contain a ds:Reference computed on the
767 SignedProperties element.

768 Below follows the schema definition for this element.

```
769 <xsd:element name="IndividualDataObjectsTimeStamp"  
770 type="XAdESTimeStampType"/>
```

771
772 This property uses the explicit (Include) mechanism. Generating applications shall compose the Include elements
773 to refer to those ds:Reference elements that are to be time-stamped. Their corresponding referencedData
774 attribute shall be present and set to "true".

775 The message imprint computation input shall be the result of processing the selected ds:Reference within
776 ds:SignedInfo as follows:

- 777 1) Process the retrieved ds:Reference element according to the reference processing model of XMLDSIG.
- 778 2) If the result is a XML node set, canonicalize it as specified in clause 5.4.
- 779 3) Concatenate the resulting octets to those resulting from previously processed ds:Reference elements in
780 ds:SignedInfo.

781 6.2.9 The SignaturePolicyIdentifier element (XAdES-EPES)

782 The SignaturePolicyIdentifier property is a signed property qualifying the signature. At most one
783 SignaturePolicyIdentifier element may be present in the signature.

784 Below follows the Schema definition for this type.

```
785 <xsd:element name="SignaturePolicyIdentifier" type="SignaturePolicyIdentifierType"/>
```

```

786 <xsd:complexType name="SignaturePolicyIdentifierType">
787   <xsd:choice>
788     <xsd:element name="SignaturePolicyId" type="SignaturePolicyIdType"/>
789     <xsd:element name="SignaturePolicyImplied"/>
790   </xsd:choice>
791 </xsd:complexType>
792
793 <xsd:complexType name="SignaturePolicyIdType">
794   <xsd:sequence>
795     <xsd:element name="SigPolicyId" type="ObjectIdentifierType"/>
796     <xsd:element ref="ds:Transforms" minOccurs="0"/>
797     <xsd:element name="SigPolicyHash" type="DigestAlgAndValueType"/>
798     <xsd:element name="SigPolicyQualifiers"
799       type="SigPolicyQualifiersListType" minOccurs="0"/>
800   </xsd:sequence>
801 </xsd:complexType>
802
803 <xsd:complexType name="SigPolicyQualifiersListType">
804   <xsd:sequence>
805     <xsd:element name="SigPolicyQualifier" type="AnyType"
806       minOccurs="unbounded"/>
807   </xsd:sequence>
808 </xsd:complexType>
809
810

```

811 The `SignaturePolicyId` element will appear when the signature policy is identified using the first alternative. The
812 `SigPolicyId` element contains an identifier that uniquely identifies a specific version of the signature policy.

813 The `SigPolicyHash` element contains the identifier of the hash algorithm and the hash value of the signature policy.

814 The optional `ds:Transforms` element may contain the transformations performed on the signature policy document
815 before computing its hash. The processing model for these transformations is described in [2].

816 The present document defines a new Transform, which will be identified by setting the `ds:Transform's`
817 `Algorithm` attribute's value to:

818 `http://uri.etsi.org/19132/v1.1.1/SignaturePolicy/SPDocDigestAsInSpecification`

819 If used, this transform notifies that the hash value of the signature policy document has been computed as specified in a
820 certain technical specification. If this transform is used, then the `SignaturePolicyIdentifier` shall be qualified
821 at least by the `SPDocSpecification` qualifier, specified in (6.2.9.1), which identifies the aforementioned technical
822 specification.

823 This transform may be used when the technical specification defines a mechanism for computing the hash value of the
824 signature policy document that is not easily implementable using widely used XML technologies (e.g. XPath), as might
825 occur, for instance, when the signature policy document is DER-encoded ASN.1.

826 This transform shall not be used in elements different than `SignaturePolicyId` element.

827 **EDITOR NOTE: feedback from stakeholders is kindly requested, as the rationalized framework of electronic**
828 **signatures anticipates that signature policy documents may be written in XML, ASN.1 or written in**
829 **human readable way, and in past interoperability test events, raised an interoperability problem when a**
830 **XAdES signature was pointing to a ASN.1 defined signature policy document.**

831 The `SigPolicyQualifier` element may contain additional information qualifying the signature policy identifier.

832 Alternatively, the `SignaturePolicyImplied` empty element indicates that the data object(s) being signed and
833 other external data imply the signature policy. It appears when the signature policy can be unambiguously derived from
834 the semantics of the type of data object(s) being signed, and some other information.

835 6.2.9.1 Signature policy qualifier

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

- 837 • a URL where a copy of the signature policy may be obtained (SPURI element);
- 838 • a user notice that should be displayed when the signature is verified (SPUserNotice element).
- 839 • An identifier of the technical specification that defines the syntax used for producing the signature policy
- 840 document (SPDocSpecification element).

841 Below follows the Schema definition for SPURI and SPUserNotice elements.

```
842 <xsd:element name="SPURI" type="xsd:anyURI"/>
843 <xsd:element name="SPUserNotice" type="SPUserNoticeType"/>
844
845 <xsd:complexType name="SPUserNoticeType">
846   <xsd:sequence>
847     <xsd:element name="NoticeRef" type="NoticeReferenceType"
848       minOccurs="0"/>
849     <xsd:element name="ExplicitText" type="xsd:string"
850       minOccurs="0"/>
851   </xsd:sequence>
852 </xsd:complexType>
853
854 <xsd:complexType name="NoticeReferenceType">
855   <xsd:sequence>
856     <xsd:element name="Organization" type="xsd:string"/>
857     <xsd:element name="NoticeNumbers" type="IntegerListType"/>
858   </xsd:sequence>
859 </xsd:complexType>
860
861 <xsd:complexType name="IntegerListType">
862   <xsd:sequence>
863     <xsd:element name="int" type="xsd:integer" minOccurs="0"
864       maxOccurs="unbounded"/>
865   </xsd:sequence>
866 </xsd:complexType>
867
```

868 The SPUserNotice element is intended for being displayed whenever the signature is validated. The
 869 ExplicitText element contains the text of the notice to be displayed. Other notices could come from the
 870 organization issuing the signature policy. The NoticeRef element names an organization and identifies by numbers
 871 (NoticeNumbers element) a group of textual statements prepared by that organization, so that the application could
 872 get the explicit notices from a notices file.

873 Below follows the Schema definition for SPDocSpecification element:

```
874 <!-- targetNamespace="http://uri.etsi.org/19132/v1.1.1#" -->
875 <xsd:element name="SPDocSpecification" type="xades:ObjectIdentifierType"/>
876
```

877 If the technical specification is identified using an OID, then the xades:Identifier child shall contain a URN
 878 encoding this OID as specified in RFC 3061 [10], and its QualifierType attribute shall be present with its value set
 879 to "OIDAsURN". If the technical specification is identified using a URI, then the xades:Identifier child shall
 880 contain this URI and its QualifierType attribute shall not be present.

881 **EDITOR NOTE:** This new qualifier will allow to identify whether the signature policy document is human readable,
 882 XML encoded, or ASN.1 encoded, by identifying the specific Technical Specifications where these
 883 formats will be defined.

884 6.2.10 The xadesenv111:SignaturePolicyStore element

885 The xadesenv111:SignaturePolicyStore is an optional unsigned property qualifying the signature.

886 The xadesenv111:SignaturePolicyStore property may be used to store the signature policy document which
 887 is referenced in the SignaturePolicyIdentifier attribute so that it can be used for offline and long-term
 888 validation.

889 Below follows the schema definition for this element:

```
890 <!-- targetNamespace="http://uri.etsi.org/19132/v1.1.1#" -->
891
892
893 <xsd:element name="SignaturePolicyStore" type="xadesenv111:SignaturePolicyStoreType"/>
894 <xsd:complexType name="SignaturePolicyStoreType">
895   <xsd:sequence>
896     <xsd:element ref="xadesenv111:SPDocSpecification"/>
897     <xsd:choice>
898       <xsd:element name="SignaturePolicyDocument" type="xsd:base64Binary"/>
899       <xsd:element name="SigPolDocLocalURI" type="xsd:anyURI"/>
900     </xsd:choice>
901   </xsd:sequence>
902 </xsd:complexType>
903
```

904 The `xadesenv111:SignaturePolicyStore` element may contain the base-64 encoded signature policy document as content of the `xadesenv111:SignaturePolicyDocument` element, or an URI to a local store where this document may be retrieved, as `xadesenv111:SigPolDocLocalURI` element's value.

907 NOTE 1: The URI value within `xadesenv111:SigPolDocLocalURI` element may be different than the SPURI qualifier's value.

909 The `xadesenv111:SPDocSpecification` element shall identify the technical specification that defines the syntax used for producing the signature policy document.

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

913 6.3 The `SignatureTimeStamp` element (XAdES-T)

914 The `SignatureTimeSamp` property is an optional unsigned property qualifying the signature. A XAdES-T form signature may contain several `SignatureTimeSamp` elements, encapsulating time-stamp tokens obtained from different TSAs

917 The `SignatureTimeStamp` element encapsulates encapsulates the time-stamp token over the `ds:SignatureValue` element.

919 Below follows the schema definition for this element.

```
920 <xsd:element name="SignatureTimeStamp" type="XAdESTimeStampType"/>
921
```

922 This property uses the implicit mechanism as the time-stamped data object is always the same. For building the input to the message imprint computation, applications shall:

- 924 1) Take the `ds:SignatureValue` element and its contents.
- 925 2) Canonicalize it as specified in clause 5.4.

926 6.4 Properties for validation data values

927 This clause describes in detail those properties that allow the incorporation of validation data values to the electronic signature.

929 6.4.1 The `CertificateValues` Property element

930 The `CertificateValues` property is an optional unsigned property qualifying the signature. There shall be at most one occurrence of this property in the signature.

932 The `CertificateValues` element contains the full set of certificates that have been used to validate the electronic signature, including the signer's certificate, except those ones already present in the `ds:KeyInfo` element of the signature.

935 Below follows the schema definition for this element.

```

936 <xsd:element name="CertificateValues" type="CertificateValuesType"/>
937
938 <xsd:complexType name="CertificateValuesType">
939   <xsd:choice minOccurs="0" maxOccurs="unbounded">
940     <xsd:element name="EncapsulatedX509Certificate"
941       type="EncapsulatedPKIDataType"/>
942     <xsd:element name="OtherCertificate" type="AnyType"/>
943   </xsd:choice>
944   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
945 </xsd:complexType>
946

```

947 The EncapsulatedX509Certificate element is able to contain the base-64 encoding of a DER-encoded X.509
 948 certificate. The OtherCertificate element is a placeholder for potential future new formats of certificates.

949 Should XML time-stamp tokens based in XMLDSIG be standardized and spread, this type could also serve to contain
 950 the certification chain for any TSUs providing such time-stamp tokens, if these certificates are not already present in the
 951 time-stamp tokens themselves as part of the TSUs' signatures. In this case, an element of this type could be added as an
 952 unsigned property to the XML time-stamp token using the incorporation mechanisms defined in the present document.

953 6.4.2 The RevocationValues property element

954 The RevocationValues property is an optional unsigned property that qualifies the signature. There shall be at
 955 most one occurrence of this property in the signature.

956 The RevocationValues property element is used to hold the values of the revocation information that are to be
 957 shipped with the electronic signature.

958 Below follows the Schema definition for this element.

```

959 <xsd:element name="RevocationValues" type="RevocationValuesType"/>
960
961 <xsd:complexType name="RevocationValuesType">
962   <xsd:sequence>
963     <xsd:element name="CRLValues" type="CRLValuesType"
964       minOccurs="0"/>
965     <xsd:element name="OCSPValues" type="OCSPValuesType"
966       minOccurs="0"/>
967     <xsd:element name="OtherValues" type="OtherCertStatusValuesType"
968       minOccurs="0"/>
969   </xsd:sequence>
970   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
971 </xsd:complexType>
972

```

973 Revocation information can include Certificate Revocation Lists (CRLValues) or responses from an online certificate
 974 status server (OCSPValues). Additionally a placeholder for other revocation information (OtherValues) is
 975 provided for future use.

```

976 <xsd:complexType name="CRLValuesType">
977   <xsd:sequence>
978     <xsd:element name="EncapsulatedCRLValue"
979       type="EncapsulatedPKIDataType"
980       maxOccurs="unbounded"/>
981   </xsd:sequence>
982 </xsd:complexType>
983

```

984 Certificate Revocation Lists (CRLValues) shall consist of a sequence of at least one Certificate Revocation List. Each
 985 EncapsulatedCRLValue shall contain the base-64 encoding of a DER-encoded X.509 CRL. Should the validation
 986 data contain one or more Delta CRLs, this property shall include the set of CRLs required to provide complete
 987 revocation lists.

```

988 <xsd:complexType name="OCSPValuesType">
989   <xsd:sequence>
990     <xsd:element name="EncapsulatedOCSPValue"
991       type="EncapsulatedPKIDataType" maxOccurs="unbounded"/>
992   </xsd:sequence>
993 </xsd:complexType>
994

```

995 OCSP Responses (OCSPValues) consist of a sequence of at least one OCSP Response. The
 996 EncapsulatedOCSPValue element shall contain the base-64 encoding of a DER-encoded OCSPResponse
 997 defined in RFC 2560 [7].

```
998 <xsd:complexType name="OtherCertStatusValuesType">
999   <xsd:sequence>
1000     <xsd:element name="OtherValue" type="AnyType"
1001       maxOccurs="unbounded"/>
1002   </xsd:sequence>
1003 </xsd:complexType>
1004
```

1005 The OtherValues element provides a placeholder for other revocation information that can be used in the future.

1006 Should XML time-stamp tokens based in XMLDSIG be standardized and spread, this type could also serve to contain
 1007 the values of revocation data including CRLs and OCSP responses for any TSUs providing such time-stamp tokens, if
 1008 they are not already present in the time-stamp tokens themselves as part of the TSUs' signatures. In this case, an element
 1009 of this type could be added as an unsigned property to the XML time-stamp token using the incorporation mechanisms
 1010 defined in the present document.

1011 6.4.3 The AttrAuthoritiesCertValues element

1012 The AttrAuthoritiesCertValues property is an optional unsigned property that qualifies the signature. There
 1013 shall be at most one occurrence of this property in the signature.

1014 This property contains the certificate values of the Attribute Authorities that have been used to validate the attribute
 1015 certificate when present in the signature. It may also contain values of the CA certificates within the AA certificate's
 1016 certification path if they are not present elsewhere in the XAdES signature.

1017 **EDITOR NOTE:** previous versions of XAdES did not make this issue clear. Feedback is requested from
 1018 stakeholders regarding the suitability of this amendment.

1019 It may also contain the certificates of signers of signed assertions present within the
 1020 xadesenv111:SignedAssertions element, and CA certificates within their certification paths if they are not
 1021 present elsewhere in the XAdES signature.

1022 **EDITOR NOTE:** this may also be the place for the material related to the signed assertions, even if the entities
 1023 that issue such signed assertions are not considered AttributeAuthorities.

1024 Below follows the Schema definition for this element.

```
1025 <xsd:element name="AttrAuthoritiesCertValues" type="CertificateValuesType"/>
1026
```

1027 Any certificate present within CertificateValues property, which has been used for validating the attribute
 1028 certificate, does not need to appear within the AttrAuthoritiesCertValues.

1029 6.4.4 The AttributeRevocationValues Property element

1030 The AttributeRevocationValues property is an optional unsigned property that qualifies the signature. There
 1031 shall be at most one occurrence of this property in the signature.

1032 This property contains the set of revocation data that have been used to validate the attribute certificate when present in
 1033 the signature, if not present anywhere else within the XAdES signature. It may also contain the set of revocation data
 1034 that have been used to validate the signatures of signed assertions within xadesenv111:SignedAssertions
 1035 element, if not present elsewhere within the XAdES signature.

1036 Below follows the Schema definition for this element.

```
1037 <xsd:element name="AttributeRevocationValues" type="RevocationValuesType"/>
1038
```

1039 Any revocation data present within RevocationValues property, which has been used for validate the attribute
 1040 certificate or the signed assertions, does not need to appear within the AttributeRevocationValues.

041 EDITOR NOTE: this may also be the place for the revocation material related to the signed assertions, even if the
 042 entities that issue such signed assertions are not considered AttributeAuthorities

043 Should the validation data contain one or more Delta CRLs, this property shall include the set of CRLs required to
 044 provide complete revocation lists

045 6.5 Properties for XAdES-A form

046 6.5.1 The `xadesv141:TimeStampValidationData` element

047 The `TimeStampValidationData` element is an optional unsigned property qualifying the signature. Several
 048 occurrences of this element may be present within a XAdES signature.

049 This element is specified to serve as an optional container for validation data required for carrying a full verification of
 050 time-stamp tokens embedded within any of the different time-stamp containers defined in the present document.

051 Below follows the schema definition for this element.

```
052
053
054 <!-- targetNamespace="http://uri.etsi.org/101903/v1.4.1#" -->
055
056 <xsd:element name="TimeStampValidationData" type="xadesv141:ValidationDataType"/>
057
058 <xsd:complexType name="ValidationDataType">
059   <xsd:sequence>
060     <xsd:element ref="xades:CertificateValues" minOccurs="0" />
061     <xsd:element ref="xades:RevocationValues" minOccurs="0" />
062   </xsd:sequence>
063   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
064   <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
065 </xsd:complexType>
066
```

067 The structure of `xades:CertificateValues` child is defined in clause 6.4.1. When present, it shall contain
 068 certificates used in the full verification of time-stamp tokens embedded in one XAdES time-stamp container. This
 069 element may contain all the certificates required for a full verification of the time-stamp tokens, but it may also contain
 070 only a part of them if the rest are present in other place of the XAdES signature (like within the time-stamp token itself,
 071 or even in other `xadesv141:TimeStampValidationData` created for other time-stamp tokens).

072 The structure of `xades:RevocationValues` child is defined in clause 6.4.2. When present, it shall contain the
 073 revocation information used in the full verification of time-stamp tokens embedded in one XAdES time-stamp
 074 container. This element may contain all the revocation information pieces (for instance CRLs or OCSP responses)
 075 required for a full verification of the time-stamp tokens, but it may also contain only a part of them if the rest are
 076 present in other place of the XAdES signature (like within the time-stamp token itself, or even in other
 077 `xadesv141:TimeStampValidationData` created for other time-stamp tokens).

078 Optional `Id` attribute allows referencing this element.

079 Optional `URI` attribute, when present, is used for referencing the time-stamp container of the time-stamp token whose
 080 validation data is contained within this element.

081 6.5.1.1 Use of `URI` attribute

082 When a XAdES signature requires to include all the validation data required for a full verification of a time-stamp token
 083 embedded in any of the following containers: `SignatureTimeStamp`, `RefsOnlyTimeStamp`,
 084 `SigAndRefsTimeStamp`, or `ArchiveTimeStamp`, and that validation data is not present in other parts of the
 085 signature, a new `xadesv141:TimeStampValidationData` element shall be created containing the missing
 086 validation data information and it shall be added as a child of `UnsignedSignatureProperties` elements
 087 immediately after the respective time-stamp token container element. Under these circumstances there is no need to use
 088 `URI` attribute as the identification of the related time-stamp token container is implicit in the relative position of both
 089 elements, the container and the `xadesv141:TimeStampValidationData` element.

090 When a XAdES requires to include all the validation data required for a full verification of a time-stamp token
 091 embedded in any of the following containers: `IndividualDataObjectsTimeStamp` or
 092 `AllDataObjectTimeStamp`, the treatment is different because first, there may be more than one signed time-stamp
 093 tokens containers, and second they are signed properties whereas the corresponding
 094 `xadesv141:TimeStampValidationData` elements are unsigned and they appear as children of different
 095 parents. Under these circumstances the URI attribute within the corresponding
 096 `xadesv141:TimeStampValidationData` element shall be present and shall be used to reference the specific
 097 signed container encapsulating time-stamp tokens whose validation data that element actually contains.

098 6.5.2 The `xadesv141:ArchiveTimeStamp` element

099 The `xadesv141:ArchiveTimeStamp` element is an optional unsigned property qualifying the signature. Several
 100 occurrences of this element may be present within a XAdES signature.

101 Below follows the schema definition for this element.

```
102 <xsd:element name="ArchiveTimeStamp" type="XAdESTimeStampType"/>
```

104 Should the XAdES signature incorporate a `CounterSignature` unsigned property, implementers should ensure that
 105 all the required material for conducting the validation of the counter-signature is incorporated to the XAdES signature
 106 before generating the first `xadesv141:ArchiveTimeStamp` property. This may be done within the counter-
 107 signature itself or within the containers available within the counter-signed XAdES signature.

108 Should a `CounterSignature` unsigned property be time-stamped by the `xadesv141:ArchiveTimeStamp`,
 109 any ulterior change of their contents (by addition of unsigned properties if the counter-signature is a XAdES signature,
 110 for instance) would make the validation of the `xadesv141:ArchiveTimeStamp`, and in consequence of the
 111 countersigned XAdES signature, fail. Implementers should, in consequence, not change the contents of the
 112 `CounterSignature` property once it has been time-stamped by the `xadesv141:ArchiveTimeStamp`.
 113 Implementors may, under these circumstances, to make use of the detached counter-signature mechanism specified in
 114 clause 6.2.7.1.

115 In addition it has to be noted that the present document allows to counter-sign a previously time-stamped
 116 countersignature with another `CounterSignature` property added to the embedding XAdES signature after the
 117 time-stamp container.

118 NOTE 1: Readers are warned that once an `xadesv141:ArchiveTimeStamp` property is added to the signature,
 119 any ulterior addition of a `ds:Object` to the signature would make the verification of such time-stamp fail.

120 Depending whether all the unsigned properties covered by the time-stamp token and the
 121 `xadesv141:ArchiveTimeStamp` property itself have the same parent or not, its contents may be different. Details
 122 are given in clauses below.

123 6.5.2.1 Not distributed case

124 When `xadesv141:ArchiveTimeStamp` and all the unsigned properties covered by its time-stamp token have the
 125 same parent, this property uses the Implicit mechanism for all the time-stamped data objects. The input to the
 126 computation of the digest value shall be built as follows:

- 127 1) Initialize the final octet stream as an empty octet stream.
- 128 2) Take all the `ds:Reference` elements in their order of appearance within `ds:SignedInfo` referencing
 129 whatever the signer wants to sign including the `SignedProperties` element. Process each one as indicated
 130 below:
 - 131 - Process the retrieved `ds:Reference` element according to the reference processing model of
 132 XMLDSIG.
 - 133 - If the result is a XML node set, canonicalize it as specified in clause 5.4.
 - 134 - Concatenate the resulting octets to the final octet stream.

- 135 3) Take the following XMLDSIG elements in the order they are listed below, canonicalize each one as specified
136 in clause 5.4, and concatenate each resulting octet stream to the final octet stream:
- 137 - The `ds:SignedInfo` element.
 - 138 - The `ds:SignatureValue` element.
 - 139 - The `ds:KeyInfo` element, if present.
- 140 4) Take the unsigned signature properties that appear before the current `xadesv141:ArchiveTimeStamp`
141 in the order they appear within the `xades:UnsignedSignatureProperties`, canonicalize each one as
142 specified in clause 5.4, and concatenate each resulting octet stream to the final octet stream. While
143 concatenating the following rules apply:
- 144 - The `xades:CertificateValues` property shall be added if it is not already present and the
145 `ds:KeyInfo` element does not contain the full set of certificates used to validate the electronic signature.
 - 146 - The `xades:RevocationValues` property shall be added if it is not already present and the
147 `ds:KeyInfo` element does not contain the revocation information that has to be shipped with the electronic
148 signature.
 - 149 - The `xades:AttrAuthoritiesCertValues` property shall be added if not already present and the
150 following conditions are true: there exist an attribute certificate in the signature AND a number of
151 certificates that have been used in its validation do not appear in `CertificateValues`.
 - 152 - The `xades:AttributeRevocationValues` property shall be added if not already present and
153 there the following conditions are true: there exist an attribute certificate AND some revocation data that
154 have been used in its validation do not appear in `RevocationValues`.
- 155 5) Take all the `ds:Object` elements except the one containing `xades:QualifyingProperties` element.
156 Canonicalize each one as specified in clause 5.4, and concatenate each resulting octet stream to the final octet
157 stream.

158 6.5.2.2 Distributed case

159 When `xadesv141:ArchiveTimeStamp` and some of the unsigned properties covered by its time-stamp token DO
160 NOT have the same parent, applications shall use the explicit (based on `xades:Include` elements) mechanism only
161 for referencing the unsigned properties. Applications SHALL build one `xades:Include` element for each unsigned
162 property that is covered by the time-stamp token. These `xades:Include` elements will be added in the same order as
163 the unsigned properties are processed for contributing to the digest computation input.

164 No `xades:Include` elements are generated for any other XMLDSIG element present in the signature, although they
165 are actually time-stamped as they contribute to the generation of the message imprint computation input.

166 Generating applications shall build digest computation input as for the Implicit case (clause 6.5.4.1) substituting step 4
167 by the one specified below:

- 168 4) Take the unsigned signature properties present in the signature, extract comment nodes, canonicalize each one
169 as specified in clause 5.4, and concatenate each resulting octet stream to the final octet stream. While
170 concatenating, the following rules apply:
- 171 - The `xades:CertificateValues` property shall be added if it is not already present and the
172 `ds:KeyInfo` element does not contain the full set of certificates used to validate the electronic signature.
 - 173 - The `xades:RevocationValues` property shall be added if it is not already present and the
174 `ds:KeyInfo` element does not contain the revocation information that has to be shipped with the electronic
175 signature.
 - 176 - The `xades:AttrAuthoritiesCertValues` property shall be added if not already present and the
177 following conditions are true: there exist an attribute certificate in the signature AND a number of
178 certificates that have been used in its validation do not appear in `CertificateValues`.

- The `xades:AttributeRevocationValues` property shall be added if not already present and the following conditions are true: there exist attribute certificates AND some revocation data that have been used in its validation do not appear in `RevocationValues`.

6.5.3 The `xadesenv111:RenewedDigests` element

EDITOR NOTE: Feedback on the suitability of this property and its definition is kindly requested to stakeholders.

The `xadesenv111:RenewedDigests` property is an optional unsigned property qualifying the signature. Several occurrences of this element may be present within a XAdES signature. This property may be used when the electronic signature contains one or more signed `ds:Manifest` referencing data objects that are detached from the signature.

The `xadesenv111:RenewedDigests` property contains the digest values of the aforementioned indirectly signed detached data objects computed with a stronger digest algorithm than the one used for computing the digest values present within the `ds:Manifest`'s `ds:Reference` children.

Below follows the schema definition for this element. Note that the elements are defined within the `xadesenv111` namespace, whose URI is <http://uri.etsi.org/19132/v1.1.1#>.

```

192 <!-- targetNamespace="http://uri.etsi.org/19132/v1.1.1#" -->
193
194 <xsd:element name="RenewedDigests" type="xadesenv111:RenewedDigestsType"/>
195
196 <xsd:complexType name="RenewedDigestsType">
197   <xsd:sequence>
198     <xsd:element ref="ds:DigestMethod"/>
199     <xsd:element name="RecomputedDigestValue" type="xadesenv111:RecomputedDigestValueType"
200 maxOccurs="unbounded"/>
201   </xsd:sequence>
202   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
203 </xsd:complexType>
204
205 <xsd:complexType name="RecomputedDigestValueType">
206   <xsd:simpleContent>
207     <xsd:extension base="ds:DigestValueType">
208       <xsd:attribute name="order" type="xsd:integer" use="required"/>
209     </xsd:extension>
210   </xsd:simpleContent>
211 </xsd:complexType>
212

```

The `ds:DigestMethod` child indicates a digest algorithm (stronger than the algorithm used for computing the digest of the signed data objects referenced within a signed `ds:Manifest` element).

Each `RecomputedDigestValue` child contains the base-64 encoded digest value, computed with the digest algorithm indicated within the aforementioned `ds:DigestMethod`, of some (or all) of the signed data objects referenced within a signed `ds:Manifest` element.

The mandatory `RecomputedDigestValue`'s `order` attribute identifies the specific `ds:Reference` element that was referencing the data object whose digest value is recomputed with the stronger digest algorithm. Its value is an integer that identifies the order of appearance of the identified `ds:Reference` when the XAdES signature is serialized, where the value "1" is assigned to the first `ds:Reference` child of the `ds:SignedInfo` element.

When implementers suspect that a certain digest algorithm is becoming weak, when one or more detached data objects have been indirectly signed using that algorithm with a signed `ds:Manifest`, and when they suspect that the aforementioned data objects might be substituted by others with the same digest due to the weakness of the digest algorithm, they may decide to generate a `xadesenv111:RenewedDigests` element including the digest values of the aforementioned data objects computed with a stronger algorithm, and incorporate it to the XAdES signature. This recomputation of digest counters the threat explained in clause D.1.14.

When validating a XAdES signature that incorporates this property, the validation process shall include a step consisting in taking each `ds:Reference` element within `xadesenv111:RenewedDigests` property, process it according to the XMLDSIG processing model, digest the retrieved data object using the digest method indicated within this `ds:Reference`, and compare it with the digest value present within this `ds:Reference`. In absence of any other failure in the validating process, a failure in one of these checks indicates that the corresponding data object indirectly signed has been replaced by another data object.

234 Figure 2 shows a XAdES signature that incorporates a signed `ds:Manifest`, whose `ds:Reference` children
235 reference two detached data objects, which is generated at time t_2 and time-stamped at time t_3 . The digest values
236 present within the aforementioned `ds:Reference` elements have been computed using a digest algorithm `alg1`.

237 The figure shows that at a certain posterior time, it is suspected that algorithm `alg1` is becoming too weak and is decided
238 to incorporate in the signature the digest on the two detached data objects computed with a stronger digest algorithm
239 `alg2`, within the `xadesenv111:RenewedDigests` unsigned property (times t_4 and t_5 respectively), and afterwards,
240 incorporate a `xadesv141:ArchiveTimeStamp` in t_6 .

241 The figure also shows that the XAdES signature includes two `ds:SignedInfo`'s `ds:Reference` children elements
242 (which would be the first and the second `ds:Reference` elements appearing within the serialized signature), and two
243 `ds:Manifest`'s `ds:Reference` children elements, which would appear respectively as third and fourth
244 `ds:Reference` elements (orders 3 and 4) when serializing the signature. In consequence, the values of the `order`
245 attributes of the `xadesenv111:RenewedDigest` elements containing the renewed digest values of the data objects
246 referenced by the two aforementioned `ds:Manifest`'s `ds:Reference` children elements, would be "3" and "4"
247 respectively.

248 Usage of `xadesenv111:RenewedDigests` unsigned property achieves two effects: first of all it counters the
249 threat resulting of the combination of digest algorithm break and detached data object substitution; and secondly it
250 allows to identify the substituted data object and preserve the validity of the signature for the not substituted detached
251 data objects. Clause D.1.14 provides additional details.

Draft

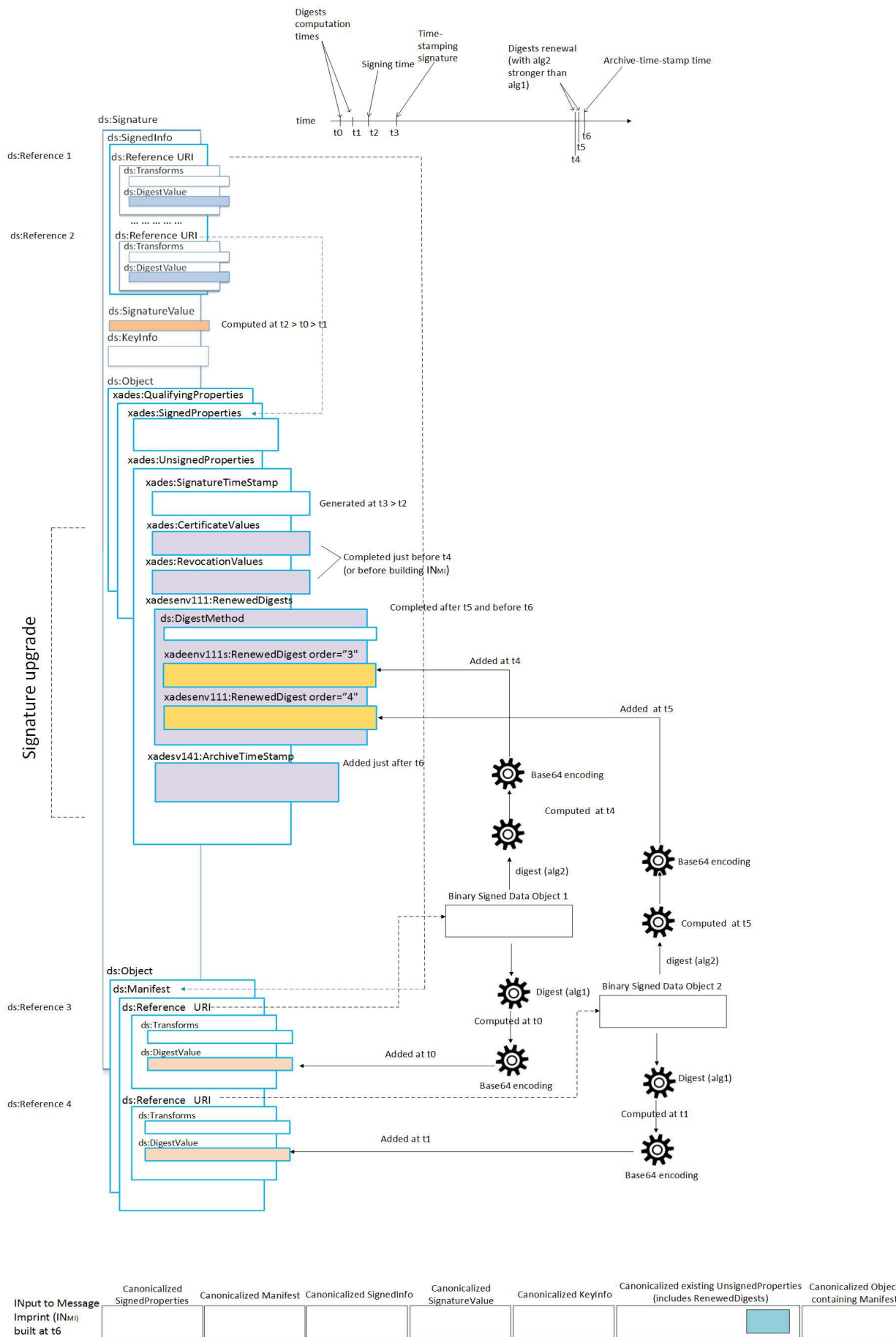


Figure 2: Use of xadesenv111:RenewedDigests

254

255

256 7. Conformance requirements

257 The present document defines several conformance levels. Subclauses below define the conformance levels for the
258 XAdES forms specified within clause 4.1 of the normative body part, namely:

- 259 • XAdES Basic Electronic Signature (XAdES-BES)
- 260 • XAdES Explicit Policy-based Electronic Signature (XAdES-EPES)
- 261 • XAdES with trusted Time (XAdES-T)
- 262 • XAdES with Archive-time-stamp (XAdES-A)

263 The normative annex C defines additional conformance levels for XAdES forms that include references to validation
264 material, namely:

- 265 • electronic signatures with Complete validation data references (XAdES-C)
- 266 • EXtended electronic signature with time forms, Type 1 and Type 2 (XAdES-X Type 1 and XAdES-X Type 2)
- 267 • EXtended Long electronic signatures with time forms, Type 1 and Type 2 (XAdES-X-L Type 1, and XAdES-
268 X-L Type 2)

269 NOTE: The conformance levels defined within the present document are more general than the levels defined in
270 the baseline profile, see part 2 of the current series.

271 An implementation claiming to be conformant to a specific level of the present document shall fulfil the corresponding
272 requirements defined in the current clause or in appendix C.

273 7.1 XAdES-Basic Electronic Signature (XAdES-BES) 274 conformance level

275 A XAdES signature claiming conformance to XAdES-BES level shall, at a minimum, consisting of the following
276 components:

- 277 • The `ds:Signature` element as specified in [2].
- 278 • At least one of the following:
 - 279 - One of the signed properties referencing the signing certificate, i.e. the `SigningCertificate` (as
280 defined in clause 6.2.2.1) or `xadesenv111:SigningCertificate` (as defined in clause 6.2.2.2)
281 incorporated (directly or indirectly) to the signature as defined in clause 5.3;
 - 282 - the `ds:KeyInfo` element whose contents satisfy the restrictions specified in clause 4.1.1.

283 7.2 XAdES-Explicitly Policy based Electronic Signature 284 (XAdES-EPES) conformance level

285 A XAdES signature claiming conformance to XAdES-EPES level shall, at a minimum, consisting of the following
286 components:

- 287 • All the components required for fulfilling conformance to XAdES-BES conformance level plus
- 288 • The `xades:SignaturePolicyIdentifier` signed property directly or indirectly incorporated.

289

7.3 XAdES with trusted Time(XAdES-T) conformance level

290

291

292

A signature claiming conformance to xAdES-T level shall be built upon a signature compliant with XAdES-BES or XAdES-EPES conformance level, and, in addition, there shall exist one or more trusted times associated with the signature. An instance of trusted time may be provided by:

293

- a `xades:SignatureTimeStamp` unsigned property directly or indirectly incorporated to the signature; or

294

- a time-mark of the electronic signature provided by a Trusted Service Provider.

295

7.4 XAdES with Archive-time-stamp (XAdES-A) conformance level

296

297

298

A XAdES signature claiming conformance to XAdES-A level shall be built upon signatures compliant with XAdES-T, XAdES-C, XAdES-X (type 1 or 2), and XAdES-XL (type 1 or 2) conformance levels. In addition it:

299

300

- shall directly or indirectly incorporate one or more instances of `xadesv141:ArchiveTimeStamp` property.

301

302

- may directly or indirectly incorporate one instance of `xades:CertificateValues` property. See clause 6.4.1 for details.

303

304

- may directly or indirectly incorporate one instance of `xades:RevocationValues` property. See clause 6.4.2 for details.

305

306

- may directly or indirectly incorporate one instance of `xades:AttrAuthoritiesCertValues`. See clause 6.4.3 for details.

307

308

- may directly or indirectly incorporate one instance of `xades:AttributeRevocationValues` property. See clause 6.4.4 for details.

309

310

311

312

Annex <A> (normative): Additional Qualifying Properties Specification

313

A.1 Qualifying properties for validation data

314

315

The following sub-clauses describe in detail qualifying properties that can contain references to certificates and revocation values that have been used in the validation of the electronic signature.

316

317

A.1.1 References to CA certificates

318

319

320

Sub-clauses below specify two unsigned properties qualifying the signature used as containers of references and digest values of CA certificates, namely: `CompleteCertificateRefs` and `xadesenv111:CompleteCertificateRefs`.

321

322

Only one of these two properties may be incorporated to a XAdES signature: if one of these properties is incorporated, then the other one shall not be incorporated.

323

324

325

326

EDITOR NOTE: feedback from stakeholders is requested on the solution proposed: define the new `xadesenv111:CompleteCertificateRefs` for acknowledging deprecation of `ds:X509IssuerSerial`, but keep `xades:CompleteCertificateRefs` as in the end, the problem is created by some XML Schema validation tools

327

A.1.1.1 The `CompleteCertificateRefs` element

328

329

The `CompleteCertificateRefs` property is an optional unsigned property that qualifies the signature. There shall be at most one occurrence of this property in the signature.

330

331

The `CompleteCertificateRefs` property carries references to the CA certificates that have been used in the validation of the electronic signature.

332

Below follows the schema definition for this element.

333

334

335

336

337

338

339

340

341

```
<xsd:element name="CompleteCertificateRefs" type="CompleteCertificateRefsType"/>
<xsd:complexType name="CompleteCertificateRefsType">
  <xsd:sequence>
    <xsd:element name="CertRefs" type="CertIDListType" />
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

342

343

The `CertRefs` element contains a sequence of `Cert` elements already defined in clause 6.2.2.1, incorporating the digest of each certificate and the issuer and serial number identifier.

344

345

346

If `CompleteCertificateRefs` and `CertificateValues` are present, all the certificates referenced in `CompleteCertificateRefs` shall be present either in the `ds:KeyInfo` element of the signature or in the `CertificateValues` element.

347

348

349

350

Should XML time-stamp tokens based in XMLDSIG be standardized and spread, this type could also serve to contain references to the certification chain for any TSUs providing such time-stamp tokens. In this case, an element of this type could be added as an unsigned property to the XML time-stamp token using the incorporation mechanisms defined in the present document.

351 A.1.1.2 The `xadesenv111:CompleteCertificateRefs` element

352 The `xadesenv111:completeCertificateRefs` property is an optional unsigned property that qualifies the
353 signature. There shall be at most one occurrence of this property in the signature.

354 The `xadesenv111:CompleteCertificateRefs` property carries references to the CA certificates that have
355 been used in the validation of the electronic signature.

356 Below follows the schema definition for this element

```
357 <!-- targetNamespace="http://uri.etsi.org/19132/v1.1.1#" -->
358
359 <xsd:element name="CompleteCertificateRefs" type="xadesenv111:CompleteCertificateRefsType"/>
360 <xsd:complexType name="CompleteCertificateRefsType">
361   <xsd:sequence>
362     <xsd:element name="CertRefs" type="xadesenv111:CertIDListType"/>
363   </xsd:sequence>
364   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
365 </xsd:complexType>
366
```

367 The `xadesenv111:CertRefs` element contains a sequence of `xadesenv111:Cert` elements already defined in
368 clause 6.2.2.2, incorporating the digest of each certificate and information (as strings) of the issuer and the serial
369 number.

370 If `xadesenv111:CompleteCertificateRefs` and `CertificateValues` are present, all the certificates
371 referenced in `xadesenv111:CompleteCertificateRefs` shall be present either in the `ds:KeyInfo` element
372 of the signature or in the `CertificateValues` element

373 NOTE: See NOTE in clause 6.2.2.2 providing rationale for the definition of this new property.

374 EDITOR NOTE: Feedback is kindly requested to stakeholders on the suitability of defining this new property

375 A.1.2 The `CompleteRevocationRefs` element

376 The `CompleteRevocationRefs` property is an optional unsigned property that qualifies the signature. There shall
377 be at most one occurrence of this property in the signature. This occurrence shall not be empty.

378 The `CompleteRevocationRefs` property will carry references to revocation values used for the validation of the
379 electronic signature.

380 Below follows the schema definition for this element.

```
381 <xsd:element name="CompleteRevocationRefs"
382   type="CompleteRevocationRefsType"/>
383
384 <xsd:complexType name="CompleteRevocationRefsType">
385   <xsd:sequence>
386     <xsd:element name="CRLRefs" type="CRLRefsType" minOccurs="0"/>
387     <xsd:element name="OCSPRefs" type="OCSPRefsType" minOccurs="0"/>
388     <xsd:element name="OtherRefs" type="OtherCertStatusRefsType"
389       minOccurs="0"/>
390   </xsd:sequence>
391   <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
392 </xsd:complexType>
393
394 <xsd:complexType name="CRLRefsType">
395   <xsd:sequence>
396     <xsd:element name="CRLRef" type="CRLRefType"
397       maxOccurs="unbounded"/>
398   </xsd:sequence>
399 </xsd:complexType>
400
401 <xsd:complexType name="CRLRefType">
402   <xsd:sequence>
403     <xsd:element name="DigestAlgAndValue"
404       type="DigestAlgAndValueType"/>
405     <xsd:element name="CRLIdentifier" type="CRLIdentifierType"
```

```

406         minOccurs="0"/>
407     </xsd:sequence>
408 </xsd:complexType>
409
410 <xsd:complexType name="CRLIdentifierType">
411     <xsd:sequence>
412         <xsd:element name="Issuer" type="xsd:string"/>
413         <xsd:element name="IssueTime" type="xsd:dateTime" />
414         <xsd:element name="Number" type="xsd:integer" minOccurs="0"/>
415     </xsd:sequence>
416     <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
417 </xsd:complexType>
418
419 <xsd:complexType name="OCSPRefsType">
420     <xsd:sequence>
421         <xsd:element name="OCSPRef" type="OCSPRefType"
422             maxOccurs="unbounded"/>
423     </xsd:sequence>
424 </xsd:complexType>
425
426 <xsd:complexType name="OCSPRefType">
427     <xsd:sequence>
428         <xsd:element name="OCSPIdentifier" type="OCSPIdentifierType"/>
429         <xsd:element name="DigestAlgAndValue"
430             type="DigestAlgAndValueType"
431             minOccurs="0"/>
432     </xsd:sequence>
433 </xsd:complexType>
434
435 <xsd:complexType name="ResponderIDType">
436     <xsd:choice>
437         <xsd:element name="ByName" type="xsd:string"/>
438         <xsd:element name="ByKey" type="xsd:base-64Binary"/>
439     </xsd:choice>
440 </xsd:complexType>
441
442 <xsd:complexType name="OCSPIdentifierType">
443     <xsd:sequence>
444         <xsd:element name="ResponderID" type="ResponderIDType"/>
445         <xsd:element name="ProducedAt" type="xsd:dateTime"/>
446     </xsd:sequence>
447     <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
448 </xsd:complexType>
449
450 <xsd:complexType name="OtherCertStatusRefsType">
451     <xsd:sequence>
452         <xsd:element name="OtherRef" type="AnyType"
453             maxOccurs="unbounded"/>
454     </xsd:sequence>
455 </xsd:complexType>

```

456 The CompleteRevocationRefs element can contain:

- 457 • sequences of references to CRLs (CRLRefs element);
- 458 • sequences of references to OCSPResponse data as defined in RFC 2560 [7] (OCSPRefs element);
- 459 • other references to alternative forms of revocation data (OtherRefs element).

460 Each element in a CRLRefs sequence (CrlRef element) references one CRL. Each reference contains:

- 461 • the digest of the entire DER encoded CRL (DigestAlgAndValue element);
- 462 • a set of data (CRLIdentifier element) including the issuer (Issuer element), the time when the CRL
463 was issued (IssueTime element) and optionally the number of the CRL (Number element).
464 CRLIdentifier element contents shall follow the rules established by XMLDSIG [2] in its clause 4.5.4.1
465 for strings representing Distinguished Names. In addition, this element can be dropped if the CRL could be
466 inferred from other information. Its URI attribute could serve to indicate where the identified CRL is archived.

467 NOTE: The number element is an optional hint helping applications to get the CRL whose digest matches the
468 value present in the reference.

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

471 Each element in an `OCSPRefs` sequence (`OcspRef` element) references one OCSP response. Each reference contains:

- 472 • a set of data (`OCSPIdentifier` element) that includes an identifier of the responder and an indication of the
473 time when the response was generated. The responder may be identified by its name, using the `Byname`
474 element within `ResponderID`. The responder may also be identified by the digest of the server's public key
475 computed as mandated in RFC 2560 [7], using the `ByKey` element. In this case the content of the `ByKey`
476 element will be the DER value of the `byKey` field defined in RFC 2560, base-64 encoded. The contents of
477 `ByName` element shall follow the rules established by XMLDSIG [2] in its clause 4.5.4.1 for strings
478 representing Distinguished Names. The generation time indication appears in the `ProducedAt` element and
479 corresponds to the "ProducedAt" field of the referenced response. The optional `URI` attribute could serve to
480 indicate where the OCSP response identified is archived;
- 481 • the digest computed on the DER encoded `OCSPResponse` defined in RFC 2560 [7], appearing within
482 `DigestAlgAndValue` element. Applications claiming alignment with the present document should include
483 the `DigestAlgAndValue` element within each `OCSPRef` element.

484 Alternative forms of validation data can be included in this property making use of the `OtherRefs` element, a
485 sequence whose items (`OtherRef` elements) can contain any kind of information.

486 If `CompleteRevocationRefs` and `RevocationValues` are present, all the revocation data referenced in
487 `RevocationRefs` shall be present either in the `ds:KeyInfo` element of the signature or in the
488 `RevocationValues` property element.

489 Should XML time-stamp tokens based in XMLDSIG be standardized and spread, this type could also serve to contain
490 references to the full set of CRL or OCSP responses that have been used to verify the certification chain for any TSUs
491 providing such time-stamp tokens. In this case, an element of this type could be added as an unsigned property to the
492 XML time-stamp token using the incorporation mechanisms defined in the present document.

493

494 A.1.3 References to certificates in the certification path of 495 Attribute Authorities certificates

496 Sub-clauses below specify two unsigned properties qualifying the signature used as containers of references and digest
497 values of certificates within the certification path of AA certificates, namely: `AttributeCertificateRefs` and
498 `xadesenv111:AttributeCertificateRefs`.

499 Only one of these two properties may be incorporated to a XAdES signature: if one of these properties is incorporated,
500 then the other one shall not be incorporated.

501 **EDITOR NOTE:** feedback from stakeholders is requested on the solution proposed: define the new
502 `xadesenv111:AttributeCertificateRefs` for acknowledging deprecation of `ds:X509IssuerSerial`,
503 but keep `xades:AttributeCertificateRefs` as in the end, the problem is created by some XML Schema
504 validation tools

505

506 A.1.3.1 The `AttributeCertificateRefs` element

507 The `AttributeCertificateRefs` property is an optional unsigned property that qualifies the signature. This
508 property may be used only when a user attribute certificate is present in the signature within the signature. There shall
509 be at most one occurrence of this property in the signature.

510 The `AttributeCertificateRefs` element will carry the references to the set of Attribute Authorities certificates
511 that have been used to validate the attribute certificates. It may also contain references to the CA certificates within the
512 certification paths of the attribute certificates, if not incorporated elsewhere within the XAdES signature.

513 **EDITOR NOTE:** previous versions of XAdES did not make this issue clear. Feedback is requested from
514 stakeholders regarding the suitability of this amendment.

515 It may also contain references to the full set of revocation data that have been used to validate the signatures of signed
 516 assertions within `xadesenv111:SignedAssertions` element, if not present elsewhere within the XAdES
 517 signature.

518 **EDITOR NOTE:** it may also contain material for signed assertions. Feedback kindly requested from
 519 stakeholders.

520 Below follows the schema definition for this element.

```
521 <xsd:element name="AttributeCertificateRefs" type="CompleteCertificateRefsType"/>
```

522 NOTE: Copies of the certificates referenced in this property may be held using the
 523 `AttrAuthoritiesCertValues` property.

524 If `AttributeCertificateRefs` and `AttrAuthoritiesCertValues` are present,
 525 `AttrAuthoritiesCertValues`, `CertificateValues`, and `ds:KeyInfo` properties shall contain all the
 526 certificates referenced in `AttributeCertificateRefs`.

528 A.1.3.2 The `xadesenv111:AttributeCertificateRefs` element

529 The `xadesenv111:AttributeCertificateRefs` property is an optional unsigned property that qualifies the
 530 signature. This property may be used only when a user attribute certificate is present in the signature within the
 531 signature. There shall be at most one occurrence of this property in the signature.

532 The `xadesenv111:AttributeCertificateRefs` element will carry the references to the set of Attribute
 533 Authorities certificates that have been used to validate the attribute certificates. It may also contain references to the CA
 534 certificates within the certification paths of the attribute certificates, if not incorporated elsewhere within the XAdES
 535 signature. It may also contain references to the certificates of signers of signed assertions present within the
 536 `xadesenv111:SignedAssertions` element, and CA certificates within their certification paths if they are not
 537 present elsewhere in the XAdES signature

538 Below follows the schema definition for this element

```
539 <!-- targetNamespace="http://uri.etsi.org/19132/v1.1.1#" -->
```

```
541 <xsd:element name="AttributeCertificateRefs" type="xadesenv111:CompleteCertificateRefsType"/>
```

542 The semantics of the elements and types defined above are identical to the elements and types defined in clause A.1.1.1,
 543 with the only exception that `ds:X509IssuerSerial` element is now optional within
 544 `xadesenv111:CertIDType` and in consequence within `xadesenv111:CertIDListType` and
 545 `xadesenv111:AttributeCertificateRefs`.

546 NOTE: See NOTE in clause 6.2.2 providing rationale for the definition of this new property

547 If `xadesenv111:AttributeCertificateRefs` and `AttrAuthoritiesCertValues` are present,
 548 `AttrAuthoritiesCertValues`, `CertificateValues`, and `ds:KeyInfo` properties shall contain all the
 549 certificates referenced in `xadesenv111:AttributeCertificateRefs`.

551 A.1.4 The `AttributeRevocationRefs` element

552 The `AttributeRevocationRefs` property is an optional unsigned property that qualifies the signature. This
 553 property may be used only when a user attribute certificate is present in the signature within the signature. There shall
 554 be at most one occurrence of this property in the signature.

555 The `AttributeRevocationRefs` property may carry the references to the full set of revocation data that have
 556 been used in the validation of the attribute certificate(s) present in the signature. It may also contain references to the
 557 full set of revocation data that have been used to validate the signatures of signed assertions within
 558 `xadesenv111:SignedAssertions` element, if not present elsewhere within the XAdES signature.

559 Below follows the schema definition for this element.

```
560 <xsd:element name="AttributeRevocationRefs" type="CompleteRevocationRefsType"/>
```

561

562 NOTE: Copies of the revocation values referenced in this property may be held using the
563 AttributeRevocationValues property.

564 If AttributeRevocationRefs and AttributeRevocationValues are present,
565 AttrAuthoritiesCertValues, CertificateValues, and ds:KeyInfo shall contain the values of all the
566 objects referenced in AttributeRevocationRefs

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

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

570 Clauses below specify containers for time-stamp tokens that cover the properties that encapsulate references to
571 validation material. Two elements are specified depending on which are the elements that are actually time-stamped,
572 namely the SigAndRefsTimeStamp and the RefsOnlyTimeStamp.

573 A.1.5.1 The SigAndRefsTimeStamp element

574 The SigAndRefsTimeStamp property is an optional unsigned property qualifying the signature. Clause B.2.1
575 proposes a XAdES form that can incorporate one or more SigAndRefsTimeStamp elements.

576 Below follows the schema definition for this element.

```
577 <xsd:element name="SigAndRefsTimeStamp" type="XAdESTimeStampType"/>
```

578

579 This property contains a time-stamp token that covers the following data objects: ds:SignatureValue element, all
580 present SignatureTimeStamp elements, CompleteCertificateRefs or
581 xadesenv111:CompleteCertificateRefs, CompleteRevocationRefs, and when present,
582 AttributeCertificateRefs or xadesenv111:AttributeCertificateRefs, and
583 AttributeRevocationRefs.

584 Depending whether all the aforementioned time-stamped unsigned properties and the SigAndRefsTimeStamp
585 property itself have the same parent or not, its contents may be different. Details are given in clauses below

586 A.1.5.1.1 Not distributed case

587 When SigAndRefsTimeStamp and all the unsigned properties covered by its time-stamp token have the same
588 parent, this property uses the Implicit mechanism. The input to the computation of the digest value shall be the result of
589 taking in order each of the data objects listed below, canonicalize each one as specified in clause 5.4, and concatenate
590 the resulting octet streams:

- 591 1) The ds:SignatureValue element.
- 592 2) Those among the following unsigned properties that appear before SigAndRefsTimeStamp, in their order
593 of appearance within the UnsignedSignatureProperties element:
 - 594 - The SignatureTimeStamp elements.
 - 595 - The CompleteCertificateRefs or xadesenv111:CompleteCertificateRefs element.
 - 596 - The CompleteRevocationRefs element.
 - 597 - The AttributeCertificateRefs or xadesenv111:AttributeCertificateRefs
598 element if this property is present.
 - 599 - The AttributeRevocationRefs element if this property is present.

500 Below follows the list -in order- of data objects that contribute to the digest computation. Elements within [] contribute
 501 in their order of appearance within the `UnsignedSignatureProperties` element, not in the order they are
 502 enumerated below:

503 `(ds:SignatureValue, [SignatureTimeStamp+, (CompleteCertificateRefs |`
 504 `xadesenv111:CompleteCertificateRefs), CompleteRevocationRefs,`
 505 `(AttributeCertificateRefs |xadesenv111:AttributeCertificateRefs)?,`
 506 `AttributeRevocationRefs?]).`

507 A.1.5.1.2 Distributed case

508 When `SigAndRefsTimeStamp` and some of the unsigned properties covered by its time-stamp token DO NOT have
 509 the same parent, applications shall build this property as indicated below:

- 510 1) No `Include` element will be added for `ds:SignatureValue`. All applications shall implicitly assume its
 511 contribution to the digest input (see below in this clause).
- 512 2) Generate one `Include` element per each unsigned property that shall be covered by the time-stamp token in
 513 the order they appear listed below:
 - 514 - The `SignatureTimeStamp` elements.
 - 515 - The `CompleteCertificateRefs` element.
 - 516 - The `CompleteRevocationRefs` or `xadesenv111:CompleteCertificateRefs` element.
 - 517 - The `AttributeCertificateRefs` or `xadesenv111:AttributeCertificateRefs`
 518 element if this property is present.
 - 519 - The `AttributeRevocationRefs` element if this property is present.

520 Applications shall build URI attributes following the rules stated in clause 6.1.4.3.1.

521 Generating applications shall build digest computation input as indicated below:

- 522 1) Initialize the final octet stream as an empty octet stream.
- 523 2) Take the `ds:SignatureValue` element and its content. Canonicalize it as specified in clause 5.4, and put
 524 the result in the final octet stream.
- 525 3) Take each unsigned property listed above in the order they have been listed above (this order shall be the same
 526 as the order the `Include` elements appear in the property). For each one extract comment nodes, canonicalize
 527 it as specified in clause 5.4, and concatenate the resulting octet string to the final octet stream.

528 Below follows the list of the data objects that contribute to the digest computation. Super index ^e means that this
 529 property is referenced using explicit mechanism, i.e. that the property contains an `Include` element that references it:

530 `(ds:SignatureValue, SignatureTimeStampe+, (CompleteCertificateRefse |`
 531 `xadesenv111:CompleteCertificateRefse), CompleteRevocationRefse,`
 532 `(AttributeCertificateRefse | xadesenv111:AttributeCertificateRefse)?,`
 533 `AttributeRevocationRefse?).`

534 A.1.5.2 The `RefsOnlyTimeStamp` element

535 The `RefsOnlyTimeStamp` property is an optional unsigned property qualifying the signature. Clause B.2.1 proposes
 536 a XAdES form that can incorporate one or more `RefsOnlyTimeStamp` elements.

537 Below follows the schema definition for this element.

538 `<xsd:element name="RefsOnlyTimeStamp" type="XAdESTimeStampType"/>`
 539

540 This property contains a time-stamp token that covers the following data objects: CompleteCertificateRefs or
 541 xadesenv111:CompleteCertificateRefs, CompleteRevocationRefs, and when present,
 542 AttributeCertificateRefs or xadesenv111:CompleteCertificateRefs, and
 543 AttributeRevocationRefs.

544 Depending whether all the aforementioned time-stamped unsigned properties and the RefsOnlyTimeStamp property
 545 itself have the same parent or not, its contents may be different. Details are given in clauses below.

546 A.1.5.2.1 Not distributed case

547 When RefsOnlyTimeStamp and all the unsigned properties covered by its time-stamp token have the same parent,
 548 this property uses the Implicit mechanism. The input to the computation of the digest value shall be the result of taking
 549 those of the unsigned properties listed below that appear before the RefsOnlyTimeStamp in their order of
 550 appearance within the UnsignedSignatureProperties element, canonicalize each one as specified in clause
 551 5.4, and concatenate the resulting octet streams:

- 552 • The CompleteCertificateRefs element.
- 553 • The CompleteRevocationRefs or xadesenv111:CompleteCertificateRefs element.
- 554 • The AttributeCertificateRefs or xadesenv111:CompleteCertificateRefs element if
 555 this property is present.
- 556 • The AttributeRevocationRefs element if this property is present.

557 Below follows the list of data objects that contribute to the digest computation:

558 ([CompleteCertificateRefs | xadesenv111:CompleteCertificateRefs),
 559 CompleteRevocationRefs, (AttributeCertificateRefs |
 560 xadesenv111:CompleteCertificateRefs)?, AttributeRevocationRefs?]).

561 A.1.5.2.2 Distributed case

562 When RefsOnlyTimeStamp and some of the unsigned properties covered by its time-stamp token DO NOT have
 563 the same parent, applications shall build this property generating one Include element per each unsigned property
 564 that shall be covered by the time-stamp token in the order they appear listed below:

- 565 • The CompleteCertificateRefs or xadesenv111:CompleteCertificateRefs element.
- 566 • The CompleteRevocationRefs element.
- 567 • The AttributeCertificateRefs or xadesenv111:CompleteCertificateRefs element if
 568 this property is present.
- 569 • The AttributeRevocationRefs element if this property is present.

570 Applications shall build URI attributes following the rules stated in clause 6.1.4.3.1.

571 Generating applications shall build digest computation input as indicated below:

- 572 1) Initialize the final octet stream as an empty octet stream.
- 573 2) Take each unsigned property listed above in the order they have been listed above (this order shall be the same
 574 as the order the Include elements appear in the property). For each one extract comment nodes, canonicalize
 575 it as specified in clause 5.4, and concatenate the resulting octet stream to the final octet stream.

576 Below follows the list -in order- of the data objects that contribute to the digest computation. Superindex^e means that
 577 this property is referenced using explicit mechanism, i.e. that the property contains a Include element that references it:

578 ((CompleteCertificateRefs^e, | xadesenv111:CompleteCertificateRefs^e),
 579 CompleteRevocationRefs^e, (AttributeCertificateRefs^e |
 580 xadesenv111:CompleteCertificateRefs^e)?, AttributeRevocationRefs^e?).

581 A.2 Obsoleted qualifying properties

583 A.2.1 The ArchiveTimeStamp element

584 The ArchiveTimeStamp element is an optional unsigned property qualifying the signature. This is an obsoleted
 585 property. Below follows the schema definition for this element.

```
586 <xsd:element name="ArchiveTimeStamp" type="XAdESTimeStampType"/>
```

587 Should the XAdES signature incorporate a CounterSignature unsigned property, implementers should ensure that
 588 all the required material for conducting the validation of the counter-signature is incorporated to the XAdES signature
 589 before generating the first ArchiveTimeStamp property. This may be done within the counter-signature itself or
 590 within the containers available within the counter-signed XAdES signature.

592 Should a CounterSignature unsigned property be time-stamped by the ArchiveTimeStamp, any ulterior
 593 change of their contents (by addition of unsigned properties if the counter-signature is a XAdES signature, for instance)
 594 would make the validation of the ArchiveTimeStamp, and in consequence of the countersigned XAdES signature,
 595 fail. Implementers should, in consequence, not change the contents of the CounterSignature property once it has
 596 been time-stamped by the ArchiveTimeStamp. Implementers may, in these circumstances, to make use of the
 597 detached counter-signature mechanism specified in clause 6.2.7.1.

598 In addition it has to be noted that the present document allows to counter-sign a previously time-stamped
 599 countersignature with another CounterSignature property added to the embedding XAdES signature after the
 700 time-stamp container.

701 Depending whether all the unsigned properties covered by the time-stamp token and the ArchiveTimeStamp
 702 property itself have the same parent or not, its contents may be different. Details are given in clauses below.

703 A.2.1.1 Not distributed case

704 When ArchiveTimeStamp and all the unsigned properties covered by its time-stamp token have the same parent,
 705 this property uses the Implicit mechanism for all the time-stamped data objects. The input to the computation of the
 706 digest value shall be built as follows:

- 707 1) Initialize the final octet stream as an empty octet stream.
- 708 2) Take all the ds:Reference elements in their order of appearance within ds:SignedInfo referencing
 709 whatever the signer wants to sign including the SignedProperties element. Process each one as indicated
 710 below:
 - 711 - Process the retrieved ds:Reference element according to the reference processing model of
 712 XMLDSIG.
 - 713 - If the result is a XML node set, canonicalize it as specified in clause 5.4.
 - 714 - Concatenate the resulting octets to the final octet stream.

- 715 3) Take the following XMLDSIG elements in the order they are listed below, canonicalize each one as specified
716 in clause 5.4, and concatenate each resulting octet stream to the final octet stream:
- 717 - The `ds:SignedInfo` element.
 - 718 - The `ds:SignatureValue` element.
 - 719 - The `ds:KeyInfo` element, if present.
- 720 4) Take any of the following unsigned signature properties that appear before the current `ArchiveTimeStamp`
721 in the order they appear within the `UnsignedSignatureProperties`, canonicalize each one as specified
722 in clause 5.4, and concatenate each resulting octet stream to the final octet stream:
- 723 - Any present `SignatureTimeStamp` property.
 - 724 - Any present `CounterSignature` property.
 - 725 - The `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs`, and
726 `CompleteRevocationRefs` properties if present.
 - 727 - The `AttributeCertificateRefs` or `xadesenv111:AttributeCertificateRefs`, and
728 `AttributeRevocationRefs` properties if present.
 - 729 - Any present `SigAndRefsTimeStamp` and `RefsOnlyTimeStamp` property.
 - 730 - The `CertificateValues` property. This property shall be added if it is not already present AND the
731 `ds:KeyInfo` element does not contain the full set of certificates used to validate the electronic
732 signature.
 - 733 - The `RevocationValues` property. This property shall be added if it is not already present AND the
734 `ds:KeyInfo` element does not contain the full set of revocation data used to validate the electronic
735 signature.
 - 736 - The `AttrAuthoritiesCertValues` property. This property shall be added if not already present
737 and the following conditions are true: there exist an attribute certificate in the signature AND a number
738 of certificates that have been used in its validation do not appear in `CertificateValues` or
739 `ds:KeyInfo`.
 - 740 - The `AttributeRevocationValues` property. This property shall be added if not already present
741 and there the following conditions are true: there exist an attribute certificate AND some revocation data
742 that have been used in its validation do not appear in `RevocationValues` or `ds:KeyInfo`.
 - 743 - Any previous `ArchiveTimeStamp` property.
- 744 5) Take any `ds:Object` element in the signature that is not referenced by any `ds:Reference` within
745 `ds:SignedInfo`, except that one containing the `QualifyingProperties` element. Canonicalize each one as
746 specified in clause 5.4, and concatenate each resulting octet stream to the final octet stream.

747 A.2.1.2 Distributed case

748 When `ArchiveTimeStamp` and some of the unsigned properties covered by its time-stamp token DO NOT have the
749 same parent, applications shall use the explicit (Include) mechanism only for referencing the unsigned properties.
750 Applications shall build this property generating one `Include` element per each unsigned property that must be
751 covered by the time-stamp token in the order they appear listed below:

- 752 1) Any present `SignatureTimeStamp` property.
- 753 2) Any present `CounterSignature` property.
- 754 3) The `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs` property if
755 present.
- 756 4) The `CompleteRevocationRefs` property if present.

- 757 5) The `AttributeCertificateRefs` or `xadesenv111:AttributeCertificateRefs` property if
758 present.
- 759 6) The `AttributeRevocationRefs` property if present.
- 760 7) Any present `SigAndRefsTimeStamp` property.
- 761 8) Any present `RefsOnlyTimeStamp` property.
- 762 9) The `CertificateValues` property. This property shall be added if it is not already present AND the
763 `ds:KeyInfo` element does not contain the full set of certificates used to validate the electronic signature.
- 764 10) `RevocationValues` property. This property shall be added if it is not already present AND the
765 `ds:KeyInfo` element does not contain the full set of revocation data used to validate the electronic
766 signature.
- 767 11) The `AttrAuthoritiesCertValues` property. This property shall be added, if not already present, when
768 the conditions mentioned in the non distributed case are met.
- 769 12) The `AttributeRevocationValues` property. This property shall be added, if not already present, when
770 the conditions mentioned in the non distributed case are met.
- 771 13) Any previously present `ArchiveTimeStamp` property.

772 No `xades:Include` elements are generated for any other XMLDSIG element present in the signature, although they
773 are actually time-stamped as they contribute to the generation of the message imprint computation input.

774 Generating applications MUST build digest computation input as for the Implicit case (clause A2.1.1) substituting step
775 4) by the one specified below:

- 776 4) Take the following unsigned signature properties in the order they are listed below, extract comment nodes,
777 canonicalize each one as specified in clause 5.4 and concatenate each resulting octet stream to the final octet
778 stream. The order of appearance of their referencing `Include` elements in the `ArchiveTimeStamp`
779 property shall be identical to the order that unsigned properties are actually processed:
- 780 - Any present `SignatureTimeStamp` property.
 - 781 - Any present `CounterSignature` property.
 - 782 - The `CompleteCertificateRefs` or `xadesenv111:CompleteCertificateRefs` property
783 if present.
 - 784 - The `CompleteRevocationRefs` property if present.
 - 785 - The `AttributeCertificateRefs` or `xadesenv111:AttributeCertificateRefs`
786 property if present.
 - 787 - The `AttributeRevocationRefs` property if present.
 - 788 - Any present `SigAndRefsTimeStamp` property.
 - 789 - Any present `RefsOnlyTimeStamp` property.
 - 790 - The `CertificateValues` property. This property shall be added if it is not already present AND the
791 `ds:KeyInfo` element does not contain the full set of certificates used to validate the electronic
792 signature.
 - 793 - The `RevocationValues` property. This property shall be added if it is not already present AND the
794 `ds:KeyInfo` element does not contain the full set of revocation data used to validate the electronic
795 signature.
 - 796 - The `AttrAuthoritiesCertValues` property. This property shall be added, if not already present,
797 when the conditions mentioned in the non distributed case are met.

- The AttributeRevocationValues property. This property shall be added, if not already present, when the conditions mentioned in the non distributed case are met.
- Any previous ArchiveTimestamp property.

Annex (normative): XAdES signature forms with references

B.1 Electronic signature with complete validation data references (XAdES-C)

XML Advanced Electronic Signature with Complete validation data references (XAdES-C) in accordance with the present document adds to the XAdES-T the CompleteCertificateRefs or xadesenv111:CompleteCertificateRefs and CompleteRevocationRefs unsigned properties as defined by the present document. If attribute certificates appear in the signature, then XAdES-C also incorporates the AttributeCertificateRefs or xadesenv111:AttributeCertificateRefs, and the AttributeRevocationRefs elements.

Below follows the structure for XAdES-C built by direct incorporation of properties on a XAdES-T containing the SignatureTimeStamp signed property. A XAdES-C form based on time-marks MAY exist without such element.

Henceforth [Ref. to certificates] denotes the choice between CompleteCertificateRefs and xadesenv111:CompleteCertificateRefs, and [Ref. to AttrAuth. certs.] denotes the choice between AttributeCertificateRefs and xadesenv111:AttributeCertificateRefs

```

817                                     XMLDISG
818                                     |
819 <ds:Signature ID?>- - - - - + - - - - - + - - - - - +
820   <ds:SignedInfo>
821     <ds:CanonicalizationMethod/>
822     <ds:SignatureMethod/>
823     (<ds:Reference URI? >
824       (<ds:Transforms>)?
825       <ds:DigestMethod/>
826       <ds:DigestValue/>
827     </ds:Reference>)+
828   </ds:SignedInfo>
829   <ds:SignatureValue/>
830   (<ds:KeyInfo>)? - - - - - +
831
832 <ds:Object>
833
834   <QualifyingProperties>
835     <SignedProperties?>
836       <SignedSignatureProperties>
837         <SignedSignatureProperties>
838           (SigningTime)?
839           ([Ref. to signing certificate])?
840           (SignaturePolicyIdentifier)?
841           (SignatureProductionPlace)?
842           ([Signer Attrs.]?)
843         </SignedSignatureProperties>
844       <SignedDataObjectProperties>
845         (DataObjectFormat)*
846         (CommitmentTypeIndication)*
847         (AllDataObjectsTimeStamp)*
848         (IndividualDataObjectsTimeStamp)*
849       </SignedDataObjectProperties>
850     </SignedProperties>
851
852
853
854

```

```

855     <UnsignedProperties>
856
857         <UnsignedSignatureProperties>
858             (CounterSignature)*- - - - - +
859             (SignatureTimeStamp)+- - - - - +
860             ([Ref. to certificates])
861             (CompleteRevocationRefs)
862             ([Ref. to AttrAuth. certs.]?)
863             (AttributeRevocationRefs)?
864         </UnsignedSignatureProperties>- - - - - +
865     </UnsignedProperties>
866
867 </QualifyingProperties>
868
869 </ds:Object>
870
871 </ds:Signature>- - - - - +
872
873
874 XAdES-BES (-EPES)
875
876 XAdES-T
877
878 XAdES-C
879

```

880 The XAdES-C form is the XAdES instantiation of the AdES-C form specified within ETSI EN 319 102.

881 Conformance requirements for this form of XAdES signatures are specified in clause C.1.

882 NOTE 1: When the signer does not provide the XAdES-C, the verifier may create the XAdES-C when the required
883 components of revocation and validation data become available. This may require a grace period.

884 B.2 Extended signatures with time forms (XAdES-X)

885 Extended signatures with time indication forms (XAdES-X) in accordance with the present document build on
886 signatures containing CompleteCertificateRefs or xadesenv111:CompleteCertificateRefs and
887 CompleteRevocationRefs properties, by adding one or more unsigned properties encapsulating time-stamp
888 tokens.

889 Depending of what is time-stamped, there are two different types of XAdES-X signatures, namely, XAdES-X type 1
890 and XAdES-X type 2. Time-stamps in both types cover, among other elements, CompleteCertificateRefs or
891 xadesenv111:CompleteCertificateRefs, and CompleteRevocationRefs properties. Time-stamps
892 provide an integrity and trusted time protection over everything that is time-stamped. They protect the referenced
893 certificates, CRLs and OCSP responses in case of a later compromise of a CA key, CRL key or OCSP issuer key.

894 XAdES-X type 2 is built by adding one or more RefsOnlyTimeStamp properties each containing one time-stamp
895 obtained from different TSAs. These time-stamps are computed on the CompleteCertificateRefs or
896 xadesenv111:CompleteCertificateRefs, and CompleteRevocationRefs properties

897 B.2.1 EXTENDED Electronic Signature with Time Type 1 (XAdES-X 898 Type 1)

899 XAdES-X type 1 is built by adding one or more SigAndRefsTimeStamp properties each containing one time-stamp
900 obtained from different TSAs. These time-stamps are computed on the SignatureValue element,
901 SignatureTimeStamp if present, CompleteCertificateRefs or
902 xadesenv111:CompleteCertificateRefs, and CompleteRevocationRefs properties. Below follows
903 the structure of this form.

```

904
905 XMLDISG
906
907 <ds:Signature ID?>- - - - - +
908 <ds:SignedInfo>
909     <ds:CanonicalizationMethod/>
910     <ds:SignatureMethod/>

```



```

910     (<ds:Reference URI? > | | | |
911       (<ds:Transforms/>)? | | | |
912       <ds:DigestMethod/> | | | |
913       <ds:DigestValue/> | | | |
914     </ds:Reference>)+ | | | |
915   </ds:SignedInfo/> | | | |
916   <ds:SignatureValue> | | | |
917   (<ds:KeyInfo>)? - - - - - +
918
919 <ds:Object>
920
921   <QualifyingProperties?>
922
923     <SignedProperties>
924
925       <SignedSignatureProperties>
926         (SigningTime)?
927         ([Ref. to signing certificate])?
928         (SignaturePolicyIdentifier)?
929         (SignatureProductionPlace)?
930         ([Signer Attrs.]?)
931       </SignedSignatureProperties>
932
933       <SignedDataObjectProperties>
934         (DataObjectFormat)*
935         (CommitmentTypeIndication)*
936         (AllDataObjectsTimeStamp)*
937         (IndividualDataObjectsTimeStamp)*
938       </SignedDataObjectPropertiesSigned>
939
940     </SignedProperties>
941
942     <UnsignedProperties>
943
944       <UnsignedSignatureProperties>
945         (CounterSignature)* - - - - - +
946         (SignatureTimeStamp)* - - - - - +
947         (([Ref. to certificates])
948         (CompleteRevocationRefs)
949         ([Ref. to AttrAuth. certs.]?)
950         (AttributeRevocationRefs)? - - - - - +
951         SigAndRefsTimeStamp +
952       </UnsignedSignatureProperties>- - - - - +
953
954     </UnsignedProperties>
955
956   </QualifyingProperties>
957
958 </ds:Object>
959 </ds:Signature>- - - - - +
960
961           XAdES-BES (-EPES) |
962
963           XAdES-T |
964
965           XAdES-C |
966
967           XAdES-X |
968

```

969 The XAdES-X-Type 1 form is the XAdES instantiation of the AdES-X-Type 1 form specified within ETSI EN 319 102.

970 Conformance requirements for this form of XAdES signatures are specified in clause C.2.1.

971

972 B.2.2 Extended Electronic Signature with Time Type 2 (XAdES-X 973 Type 2)

974 XAdES-X type 2 is built by adding one or more RefsOnlyTimeStamp properties each containing one time-stamp
975 obtained from different TSAs. These time-stamps are computed on the CompleteCertificateRefs or
976 xadesenv111:CompleteCertificateRefs, and CompleteRevocationRefs properties.

```

977                                     XMLDISG
978                                     |
979 <ds:Signature ID?>- - - - - + - - - - - + + + + +
980   <ds:SignedInfo> |
981     <ds:CanonicalizationMethod/> |
982     <ds:SignatureMethod/> |
983     (<ds:Reference URI? > |
984       (<ds:Transforms/>)? |
985       <ds:DigestMethod/> |
986       <ds:DigestValue/> |
987     </ds:Reference>)+ |
988 </ds:SignedInfo/> |
989 <ds:SignatureValue> |
990 (<ds:KeyInfo>)? - - - - - +
991
992 <ds:Object>
993
994   <QualifyingProperties>
995
996     <SignedProperties?>
997
998       <SignedSignatureProperties>
999         (SigningTime)?
1000         ([[Ref. to signing certificate]])?
1001         (SignaturePolicyIdentifier)?
1002         (SignatureProductionPlace)?
1003         ([Signer Attrs.]?)
1004       </SignedSignatureProperties>
1005
1006       <SignedDataObjectProperties>
1007         (DataObjectFormat)*
1008         (CommitmentTypeIndication)*
1009         (AllDataObjectsTimeStamp)*
1010         (IndividualDataObjectsTimeStamp)*
1011       </SignedDataObjectPropertiesSigned>
1012     </SignedProperties>
1013
1014     <UnsignedProperties>
1015
1016       <UnsignedSignatureProperties>
1017         (CounterSignature)*- - - - - +
1018         (SignatureTimeStamp)*- - - - - +
1019         ([[Ref. to certificates]])
1020         (CompleteRevocationRefs)
1021         ([Ref. to AttrAuth. certs.]?)
1022         (AttributeRevocationRefs)? - - - - - +
1023         RefsOnlyTimeStamp +
1024       </UnsignedSignatureProperties>- - - - - + + + + +
1025
1026     </UnsignedProperties>
1027
1028   </QualifyingProperties>
1029
1030 </ds:Object>
1031 </ds:Signature>- - - - - + + + + +
1032
1033                                     XAdES-BES (-EPES) |
1034                                     |
1035                                     XAdES-T |
1036                                     |
1037                                     XAdES-C |
1038                                     |
1039                                     XAdES-X |
1040
1041

```

1042 The XAdES-X-Type 2 form is the XAdES instantiation of the AdES-X-Type 2 form specified within ETSI EN 319 102.

1043 Conformance requirements for this form of XAdES signatures are specified in clause C.2.2.

B.3 Extended long electronic signatures with time forms (XAdES-X-L type 1 or 2)

Extended long electronic signatures with time (XAdES-X-L) forms in accordance with the present document build up on XAdES-X types 1 or 2 by adding the CertificateValues and RevocationValues unsigned properties.

The structure for the most complete XAdES-X-L type 1, built on the most complete XAdES-X signature, is shown below.

```

050                                     XMLDISG
051                                     |
052 <ds:Signature ID?>- - - - - +-----+-----+-----+
053   <ds:SignedInfo> |
054     <ds:CanonicalizationMethod/> |
055     <ds:SignatureMethod/> |
056     (<ds:Reference URI? > |
057       (<ds:Transforms/>)? |
058       <ds:DigestMethod/> |
059       <ds:DigestValue/> |
060     </ds:Reference>)+ |
061 </ds:SignedInfo> |
062 <ds:SignatureValue/> |
063 (<ds:KeyInfo>)? - - - - - +-----+-----+
064 |
065 <ds:Object> |
066 |
067   <QualifyingProperties> |
068 |
069     <SignedProperties?> |
070 |
071       <SignedSignatureProperties> |
072         (SigningTime)? |
073         (([Ref. to signing certificate])? |
074         (SignaturePolicyIdentifier)? |
075         (SignatureProductionPlace)? |
076         ([Signer Attrs.]? |
077       </SignedSignatureProperties> |
078 |
079       <SignedDataObjectProperties> |
080         (DataObjectFormat)* |
081         (CommitmentTypeIndication)* |
082         (AllDataObjectsTimeStamp)* |
083         (IndividualDataObjectsTimeStamp)* |
084       </SignedDataObjectPropertiesSigned> |
085 |
086     </SignedProperties> |
087 |
088     <UnsignedProperties> |
089 |
090       <UnsignedSignatureProperties> |
091         (CounterSignature)*- - - - - +-----+
092         (SignatureTimeStamp)*- - - - - +-----+
093         (([Ref. to certificates]) |
094         (CompleteRevocationRefs) |
095         ([Ref. to AttrAuth. certs.]? |
096         (AttributeRevocationRefs)? - - - - - +-----+
097         SigAndRefsTimeStamp + - - - - - +-----+
098         (CertificatesValues) |
099         (RevocationValues) |
100         (AttrAuthoritiesCertValues)? |
101         (AttributeRevocationValues)? |
102       </UnsignedSignatureProperties>- - - - - +-----+
103 |
104     </UnsignedProperties> |
105 |
106   </QualifyingProperties> |
107 |
108 </ds:Object> |
109 </ds:Signature>- - - - - +-----+-----+
110 |
111                                     XAdES-BES (-EPES) |
112 |
113                                     XAdES-T |
114 |

```

115 XAdES-C | |
 116 | |
 117 XAdES-X | |
 118 | |
 119 XAdES-X-L | |
 120

121 The structure for the most complete XAdES-X-L type 2, built on the most complete XAdES-X signature, is shown
 122 below.

```

123                                     XMLDISG
124                                     |
125 <ds:Signature ID?>- - - - -+-----+-----+-----+
126   <ds:SignedInfo>
127     <ds:CanonicalizationMethod/>
128     <ds:SignatureMethod/>
129     (<ds:Reference URI? >
130       (<ds:Transforms/>)?
131       <ds:DigestMethod/>
132       <ds:DigestValue/>
133     </ds:Reference>)+
134   </ds:SignedInfo>
135   <ds:SignatureValue/>
136   (<ds:KeyInfo>)? - - - - -+
137
138 <ds:Object>
139
140   <QualifyingProperties>
141
142     <SignedProperties>
143
144       <SignedSignatureProperties>
145         (SigningTime)?
146         (([Ref. to signing certificate])?
147         (SignaturePolicyIdentifier)?
148         (SignatureProductionPlace)?
149         ([Signer Attrs.]?
150       </SignedSignatureProperties>
151
152       <SignedDataObjectProperties>
153         (DataObjectFormat)*
154         (CommitmentTypeIndication)*
155         (AllDataObjectsTimeStamp)*
156         (IndividualDataObjectsTimeStamp)*
157       </SignedDataObjectPropertiesSigned>
158
159     </SignedProperties>
160
161     <UnsignedProperties>
162
163       <UnsignedSignatureProperties>
164         (CounterSignature)*- - - - -+
165         (SignatureTimeStamp)*- - - - -+
166         (([Ref. to certificates])
167         (CompleteRevocationRefs)
168         ([Ref. to AttrAuth. certs.]?)
169         (AttributeRevocationRefs)? - - - - -+
170         RefsOnlyTimeStamp + - - - - -+
171         (CertificatesValues)
172         (RevocationValues)
173         (AttrAuthoritiesCertValues)?
174         (AttributeRevocationValues)?
175       </UnsignedSignatureProperties>- - - - -+
176
177     </UnsignedProperties>
178
179   </QualifyingProperties>
180
181 </ds:Object>
182 </ds:Signature>- - - - -+-----+-----+
183
184                                     XAdES-BES (-EPES) |
185                                     |
186                                     XAdES-T |
187                                     |
188                                     XAdES-C |
189                                     |
190                                     XAdES-X |
  
```

191
192
193|
XAdES-X-L194
195

The XAdES-X-L-Type 1 and XAdES-X-L-Type 2 forms are the XAdES instantiations of the AdES-X-L-Type 1 and AdES-X-L-Type 2 forms specified within ETSI EN 319 102.

196

Conformance requirements for these forms of XAdES signatures are specified in clause C.3.1 and C.3.2 respectively.

197

B.4 Archival Electronic Signature complete

198

Below follows the structure of a XAdES-A built on a XAdES-X-L, as an example of the most complete archival form.

199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259

```

XMLDISG
<ds:Signature ID?>- - - - - + + + + +
  <ds:SignedInfo>
    <ds:CanonicalizationMethod/>
    <ds:SignatureMethod/>
    (<ds:Reference (URI=)? >
      (<ds:Transforms/>)?
      <ds:DigestMethod/>
      <ds:DigestValue/>
    </ds:Reference>)+
  </ds:SignedInfo>
  <ds:SignatureValue/>
  (<ds:KeyInfo>)? - - - - - +
  <ds:Object>

  <QualifyingProperties>
    <SignedProperties>
      <SignedSignatureProperties>
        (SigningTime)?
        (([Ref. to signing certificate])?
        (SignaturePolicyIdentifier)?
        (SignatureProductionPlace)?
        ([Signer Attrs.]?
        </SignedSignatureProperties>
      <SignedDataObjectProperties>
        (DataObjectFormat)*
        (CommitmentTypeIndication)*
        (AllDataObjectsTimeStamp)*
        (IndividualDataObjectsTimeStamp)*
      </SignedDataObjectPropertiesSigned>
    </SignedProperties>
    <UnsignedProperties>
      <UnsignedSignatureProperties>
        (xadesv141:TimeStampValidationData?)
        (CounterSignature)*- - - - - + + + + +
        ((SignatureTimeStamp) - - - - - + | | |
        xadesv141:TimeStampValidationData?)+ + + + +
        (([Ref. to certificates])
        (CompleteRevocationRefs)- - - - - + + + + +
        ([Ref. to AttrAuth. certs.]? | | |
        (AttributeRevocationRefs)? - - - - - + | | |
        ((SigAndRefsTimeStamp) |
        (RefsOnlyTimeStamp)) - - - - - + + + + +
        xadesv141:TimeStampValidationData?)+
        (CertificatesValues- - - - - +
        (RevocationValues
        (AttrAuthoritiesCertValues)?
        (AttributeRevocationValues)?- - - - - +
        xadesv141:ArchiveTimeStamp
        (xadesv141:TimeStampValidationData?)

```

This element, if present, contains validation data of AllDataObjectsTimeStamp or IndividualDataObjectsTimeStamp.

This element, if present, contains validation data of SignatureTimeStamp

This element, if present, contains validation data of SigAndRefsTimeStamp or RefsOnlyTimeStamp

-> First archive time-stamp

-> Each time that a new archive time-stamp

```

260         xadesv141:ArchiveTimeStamp ) *
261
262
263         </UnsignedSignatureProperties> - - - + + + + +
264
265         </UnsignedProperties>
266
267         </QualifyingProperties>
268
269     </ds:Object>
270
271 </ds:Signature> - - - - - + + + + +
272
273         XAdES-BES (-EPES)
274
275         XAdES-T
276
277         XAdES-C
278
279         XAdES-X
280
281         XAdES-X-L
282
283         XAdES-A
284

```

validation material of the former one
may be added

285

286 Annex <C> (normative):

287 Conformance requirements for additional Electronic

288 Signature Forms.

289 In addition to the base conformance levels defined in clause 7, the present document defines the following additional
290 core conformance levels:

- 291 • Electronic signatures with Complete validation data references (XAdES-C)
- 292 • EXtended electronic signature with time forms, Type 1 (XAdES-X Type 1)
- 293 • EXtended signature with time forms, Type 2 (XAdES-X Type 2)
- 294 • EXtended Long signature (XAdES-X-L)
- 295 • EXtended Long signatures with time forms, Type 1 (XAdES-X-L Type 1)
- 296 • EXtended Long signatures with time forms, Type 2 (XAdES-X-L Type 2)

297 An implementation claiming to be conformant to one of them shall implement the corresponding.

298 C.1 Electronic signatures with Complete validation data

299 references (XAdES-C)

300 A signature claiming conformance to XAdES-C level shall be built upon a signature compliant with XAdES-T
301 conformance level. In addition, it:

- 302 • shall incorporate (directly or indirectly) one instance of `xades:CompleteCertificateReferences`
303 unsigned property.
- 304 • shall incorporate (directly or indirectly) one instance of `xades:CompleteRevocationRefs` unsigned
305 property.
- 306 • may incorporate (directly or indirectly) one instance of `xades:AttributeCertificateReferences`
307 unsigned property.

- may incorporate (directly or indirectly) one instance of `xades:AttributeRevocationRefs` unsigned property.

C.2 EXTENDED SIGNATURES WITH TIME FORMS (XAdES-X)

The extended signatures with time forms build on XAdES-C and protect the certificate and revocation references with a time-stamp.

C.2.1 XAdES-X Type 1

A signature claiming conformance to XAdES-X Type 1 level shall be built upon a signature compliant with XAdES-C conformance level. In addition, it shall directly or indirectly incorporate:

- one or more instance of `xades:SigAndRefsTimeStamp` unsigned property.

C.2.2 XAdES-X Type 2

A signature claiming conformance to XAdES-X Type 2 level shall be built upon a signature compliant with XAdES-C conformance level. In addition, it shall directly or indirectly incorporate:

- one or more instance of `xades:RefsOnlyTimeStamp` unsigned property.

C.3 EXTENDED LONG SIGNATURES WITH TIME FORMS (XAdES-X-L)

C.3.1 EXTENDED LONG SIGNATURES WITH TIME FORMS, TYPE 1 (XAdES-X-L Type 1)

A signature claiming conformance to XAdES-X-L Type 1 level shall be built upon a signature having XAdES-X Type 1 level. In addition it shall incorporate the full set of certificate values and revocation values required for validating the signature. In consequence it:

- may directly or indirectly incorporate one instance of `xades:CertificateValues` property. See clause XXX for details.
- may directly or indirectly incorporate one instance of `xades:RevocationValues` property. See clause XXX for details.
- may directly or indirectly incorporate one instance of `xades:AttrAuthoritiesCertValues`. See clause XXX for details.
- may directly or indirectly incorporate one instance of `xades:AttributeRevocationValues` property. See clause XXX for details.

C.3.2.1 EXTENDED LONG SIGNATURES WITH TIME FORMS, TYPE 2 (XAdES-X-L Type 2)

A signature claiming conformance to XAdES-X-L Type 2 level shall be built upon a signature having XAdES-X Type 2 level. In addition it shall incorporate the full set of certificate values and revocation values required for validating the signature. In consequence it:

- may directly or indirectly incorporate one instance of `xades:CertificateValues` property. See clause XXX for details.
- may directly or indirectly incorporate one instance of `xades:RevocationValues` property. See clause XXX for details.

- 345 • may directly or indirectly incorporate one instance of `xades:AttrAuthoritiesCertValues`. See
346 clause XXX for details.
- 347 • may directly or indirectly incorporate one instance of `xades:AttributeRevocationValues` property.
348 See clause XXX for details.

349 Annex <D> (informative): 350 General Description

351 D.1 The `SigningTime` element

352 By adding this signed property to the XAdES signature, the signatory claims to have generated this signature at the
353 indicated time. Readers are reminded that this is not a time indication coming from a trusted source like a time-stamp
354 token.

355 D.2 References to the signing certificate

356 In many real life environments users will be able to get from different CAs or even from the same CA, different
357 certificates containing the same public key for different names. The prime advantage is that a user can use the same
358 private key for different purposes. Multiple use of the private key is an advantage when a smart card is used to protect
359 the private key, since the storage of a smart card is always limited. When several CAs are involved, each different
360 certificate may contain a different identity, e.g. as a national or as an employee from a company. Thus when a private
361 key is used for various purposes, the certificate is needed to clarify the context in which the private key was used when
362 generating the signature. Where there is the possibility of multiple uses of private keys it is necessary for the signer to
363 indicate to the verifier the precise certificate to be used.

364 Many current schemes simply add the certificate after the signed data and thus are subject to various substitution
365 attacks. An example of a substitution attack is a "bad" CA that would issue a certificate to someone with the public key
366 of someone else. If the certificate from the signer was simply appended to the signature and thus not protected by the
367 signature, any one could substitute one certificate by another and the message would appear to be signed by some one
368 else. In order to counter this kind of attack, the identifier of the certificate has to be protected by the digital signature
369 from the signer.

370 The `SigningCertificate` and `xadesenv111:SigningCertificate` properties are designed to prevent the
371 simple substitution of the certificate.

372 D.3 The `CommitmentTypeIndication` element

373 The commitment type can be indicated in the electronic signature either:

- 374 • explicitly using a commitment type indication in the electronic signature;
- 375 • implicitly or explicitly from the semantics of the signed data object.

376 If the indicated commitment type is explicit by means of a commitment type indication in the electronic signature,
377 acceptance of a verified signature implies acceptance of the semantics of that commitment type. The semantics of
378 explicit commitment types indications are specified either as part of the signature policy or may be registered for
379 generic use across multiple policies.

380 The commitment type may be:

- 381 • defined as part of the signature policy, in which case the commitment type has precise semantics that is
382 defined as part of the signature policy;
- 383 • a registered type, in which case the commitment type has precise semantics defined by registration, under the
384 rules of the registration authority. Such a registration authority may be a trading association or a legislative
385 authority.

386 The definition of a commitment type includes an identifier (URI) and an optional sequence of qualifiers, which may
387 provide additional information (for instance information about the context, be it contractual/legal/application specific).

388 If an electronic signature does not contain a recognized commitment type then the semantics of the electronic signature
389 is dependent on the data object being signed and the context in which it is being used. How commitment is indicated
390 using the semantics of the data object being signed is outside the scope of the present document.

391 D.4 The DataObjectFormat element

392 When presenting signed data to a human user it may be important that there is no ambiguity as to the presentation of the
393 signed data object to the relying party. In order for the appropriate representation (text, sound or video) to be selected
394 by the relying party a content hint may be indicated by the signer. If a relying party system does not use the format
395 specified to present the data object to the relying party, the electronic signature may not be valid. Such behaviour may
396 have been established by the signature policy, for instance.

397 The DataObjectFormat element provides information that describes the format of the signed data object. The
398 presence of this element is indeed beneficial when the signed data is to be presented to human users on validation if the
399 presentation format is not implicit within the data that has been signed.

400 D.5 The SignatureProductionPlace element

401 In some transactions the purported place where the signer was at the time of signature creation may need to be
402 indicated. In order to provide this information the SignatureProductionPlace signed property is defined. It
403 specifies an address associated with the signer at a particular geographical (e.g. city) location.

404 D.6 The SignerRole element

405 While the name of the signer is important, the position of the signer within a company or an organization is of
406 paramount importance as well. Some information (i.e. a contract) may only be valid if signed by a user in a particular
407 role, e.g. a Sales Director. In many cases, who the sales Director really is, is not that important, but being sure that the
408 signer is empowered by his company to be the Sales Director is fundamental.

409 The present document defines two different ways for providing this feature:

- 410 • by placing a *claimed* signer attribute / role;
- 411 • by placing an assertion that includes signer attribute / role, signed by an entity which does not satisfy the
412 requirements needed to be considered an Attribute Authority or
- 413 • by placing an attribute certificate issued by an Attribute Authority.

414 NOTE: Another possible approach would have been to use additional attributes containing the attribute or roles
415 name(s) in the signer's identity certificate. However, it was decided not to follow this approach as it
416 significantly complicates the management of certificates. For example, by using separate certificates for
417 the signer's identity and roles means new identity keys need not be issued if a user's role changes.

418

419 D.6.1 Claimed signer attribute/role

420 The signer may be trusted to state his own attribute or role without any third party certifying this claim; in which case,
421 the claimed role can be added to the signature as a signed attribute.

422 D.6.2 Certified signer attribute/role

423 A certified signer attribute / role is certified by a trusted Attribute Authority (AA). The certification is done either by the
424 AA issuing an Attribute.

425 Unlike public key certificates that bind an identifier to a public key, Attribute Certificates bind the identifier of a
426 certificate to some attributes, like a role. An Attribute Certificate is NOT issued by a CA but by an Attribute Authority
427 (AA). The Attribute Authority, in most cases, might be under the control of an organization or a company that is best
428 placed to know which attributes are relevant for which individual. The Attribute Authority may use or point to public
429 key certificates issued by any CA, provided that the appropriate trust may be placed in that CA. Attribute Certificates
430 may have various periods of validity. That period may be quite short, e.g. one day. While this requires that a new
431 Attribute Certificate be obtained every day, valid for that day, this can be advantageous since revocation of such
432 certificates may not be needed. When signing, the signer will have to specify which Attribute Certificate it selects. In
433 order to do so, the Attribute Certificate will have to be included in the signed data in order to be protected by the digital
434 signature from the signer. In order to unambiguously identify the attribute certificate(s) to be used for the verification of
435 the signature, an identifier of the attribute certificate(s) from the signer is part of the signed data

436 D.7 Multiple signatures and countersignatures

437 Some electronic documents have full effect only if they bear more than one signature. This is generally the case, for
438 example, when a contract is signed between two parties. The ordering of the signatures may or may not be important,
439 i.e. one may or may not need to be applied before the other. This allows establishing two basic categories for multiple
440 signatures:

- 441 • independent signatures;
- 442 • countersignatures.

443 Independent signatures are parallel signatures where the ordering of the signatures is not important. The computation of
444 these signatures is performed on exactly the same input but using different private keys.

445 Countersignatures are signatures that are applied one after the other and are used where the order the signatures are
446 applied is important. In these situations the first signature signs the signed data object. Each additional signature may
447 sign in turn the latest previously generated signature, or all the previously generated signatures and the signed
448 document.

449 The referencing mechanism present in XMLDSIG gives full support to countersignatures. Using them, the
450 countersignatures may be placed and kept in different ways: they may be embedded one within the other, or they may
451 be detached from the rest as long as their corresponding `<ds:Reference>` elements ensure that each signature
452 actually signs the previously generated signature (or all the previously generated signatures and the signed document if
453 this is the requirement). While XMLDSIG supports these features, it does not propose any standard format for
454 countersignatures as it considers this topic being out of its scope.

455 The present document defines a new URI value, which, when assigned as value of the `Type` attribute of a
456 `ds:Reference` element, denotes that the enclosing XAdES signature is in fact, a countersignature of another
457 signature.

458 In addition the present document defines with the `CounterSignature` property a standard way of managing
459 countersignatures that:

- 460 • are computed on the values of the latest previously generated signatures;
- 461 • are embedded within the signatures that they countersign so that the first electronic signature (the one
462 computed on the data objects actually signed) contains all the additional countersignatures that have to be
463 verified.

464 Independent signatures do not appear as `CounterSignature` properties of another independent one.

465 This proposal does not, of course, satisfy all the potential requirements that real situations may pose in terms of
466 relationships among electronic signatures and documents. This would require more complexity, which is out of scope of
467 XAdES. Readers are reminded, though, that ASiC containers specify a standard way of packaging together documents
468 and their detached signatures (be them independent or countersignatures).

469 D.8 Time-stamps on the signed data objects

470 The signed properties `IndividualDataObjectsTimeStamp` and `AllDataObjectsTimeStamp` provide
471 proof of the existence of the signed data object, at the time indicated by the encapsulated time-stamp token.

472 Using them, a trusted secure time may be obtained before the data objects are signed and included under the digital
473 signature. This solution requires an online connection to a trusted time-stamping service before generating the signature
474 and may not represent the precise signing time, since it can be obtained in advance. However, these properties may be
475 used by the signer to prove that the signed data object existed before the date included in the time-stamp token (see
476 clause 6.2.8).

477 D.9 The `SignaturePolicyIdentifier` element

478 The signature policy is a set of rules for the creation and validation of an electronic signature, under which the signature
479 can be determined to be valid. A given legal/contractual context may recognize a particular signature policy as meeting
480 its requirements.

481 When a comprehensive signature policy used by the verifier is either explicitly indicated by the signer or implied by the
482 data being signed, then a consistent result can be obtained when validating an electronic signature. When the signature
483 policy being used by the verifier is neither indicated by the signer nor can be derived from other data, or the signature
484 policy is incomplete then verifiers, including arbitrators, may obtain different results when validating an electronic
485 signature. Therefore, comprehensive signature policies that ensure consistency of signature validation are valuable for
486 both the signers and verifiers.

487 For assessing whether a signature policy meets the requirements of the legal and contractual context in which it is being
488 applied, a human readable form of such a policy is required. Additionally, if the parts of the signature policy that
489 specify the rules for the creation and validation of the electronic signature are expressed in a form that a computer may
490 understand and process, this will facilitate the automatic processing of an electronic signature.

491 If no signature policy is identified then the signature may be assumed to have been generated/verified without any
492 policy constraints, and hence may be given no specific legal or contractual significance through the context of a
493 signature policy.

494 The present document specifies two unambiguous ways for identifying the signature policy that a signature follows:

- 495 • The electronic signature can contain an explicit and unambiguous identifier of a signature policy together with
496 a hash value of the signature policy, so it can be verified that the policy selected by the signer is the one being
497 used by the verifier. An explicit signature policy has a globally unique reference, which, in this way, is bound
498 to an electronic signature by the signer as part of the signature calculation. In these cases, for a given explicit
499 signature policy there shall be one definitive form that has a unique binary encoded value. A signature policy
500 identified in this way may be qualified by additional information.
- 501 • Alternatively, the electronic signature can avoid the inclusion of the aforementioned identifier and hash value.
502 This will be possible when the signature policy can be unambiguously derived from the semantics of the type
503 of data object(s) being signed, and some other information, e.g. national laws or private contractual
504 agreements, that mention that a given signature policy has to be used for this type of data content. In such
505 cases, the signature will contain a specific empty element indicating that this implied way to identify the
506 signature policy is used instead the identifier and hash value.

507 D.10 The `SignatureTimeStamp` element

508 An important property for long standing signatures is that a signature, having been found once to be valid, will continue
509 to be so months or years later. A signer, verifier or both may be required to provide on request, proof that an electronic
510 signature was created or validated during the validity period of all the certificates that make up the certificate path. In
511 this case, the signer, verifier or both will also be required to provide proof that all the user and CA certificates used
512 were not revoked when the signature was created or validated.

513 It would be quite unacceptable to consider a signature as invalid even if the keys or certificates were only compromised
514 later. Thus there is a need to be able to demonstrate that the signature key was valid around the time that the signature
515 was created to provide long term evidence of the validity of a signature. Time-stamping by a Time-Stamping Authority
516 (TSA) can provide such evidence.

517 Time-stamping an electronic signature before the revocation of the signer's private key and before the end of the
518 validity of the certificate provides evidence that the signature has been created while the certificate was valid and before
519 it was revoked.

520 If a recipient wants to keep the result of the validation of an electronic signature valid, he will have to ensure that he has
521 obtained a valid time-stamp for it, before that key (and any key involved in the validation) is revoked. The sooner the
522 time-stamp is obtained after the signing time, the better. It is important to note that signatures may be generated
523 "off-line" and time-stamped at a later time by anyone, for example by the signer or any recipient interested in the
524 signature. The time-stamp can thus be provided by the signer together with the signed data object, or obtained by the
525 recipient following receipt of the signed data object.

526 The validation mandated by the signature policy can specify a maximum acceptable time difference which is allowed
527 between the time indicated in the `SigningTime` element and the time indicated by the `SignatureTimeStamp`
528 element.

529 D.11 Elements containing references to validation data

530 When dealing with long term electronic signatures, all the data used in the validation (namely, certificate path and
531 revocation information) of such signatures have to be stored and conveniently time-stamped for arbitration purposes.
532 Similar considerations apply to attribute certificates if they appear within the signature.

533 In some environments, it can be convenient to add these data to the electronic signature (as unsigned properties) for
534 archival purposes (see archival electronic signatures XAdES-A).

535 Systems implementing the present document may alternatively consider to archive validation data outside the XAdES
536 e.g. to prevent redundant storage and to reduce the size of the signatures. In such cases each electronic signature
537 incorporates references to all these data within the signature, reducing accordingly the size of the stored electronic
538 signature. This format builds up taking XAdES-T signature by incorporating additional data required for validation:

- 539 • the sequence of references to the full set of CA certificates that have been used to validate the electronic
540 signature up to (but not including) the signer's certificate;
- 541 • the sequence of references to the full set of revocation data that have been used in the validation of the signer
542 and CA certificates;
- 543 • the references to the full set of Attribute Authorities and the CA certificates that have been used to validate the
544 attribute certificate(s), if present;
- 545 • the references to the full set of revocation data that have been used in the validation of the attribute
546 certificate(s), if present.

547 The full set of references to the revocation data that have been used in the validation of the signer and CAs certificates,
548 provide means to retrieve the actual revocation data archived elsewhere in case of dispute and, in this way, to illustrate
549 that the verifier has taken due diligence of the available revocation information.

550 Currently two major types of revocation data are managed in most of the systems, namely CRLs and responses of
551 on-line certificate status servers, obtained through protocols designed for these purposes, like OCSP protocol. In
552 consequence, means are provided in the present document for referencing both types of revocation data.

553 D.12 Time-stamps on references to validation data

554 Electronic signatures incorporating time-stamps on validation data references are needed when there is a requirement to
555 safeguard against the possibility of a CA key in the certificate chain ever being compromised. A verifier may be
556 required to provide, on request, proof that the certification path and the revocation information used at the time of the
557 signature were valid, even in the case where one of the issuing keys or OCSP responder keys is later compromised.

558 The present document defines two ways of using time-stamps to protect against this compromise:

- 559 • Time-stamp the sequence formed by the digital signature (`ds:SignatureValue` element), the
560 `SignatureTimeStamp` element when present in the XAdES-T form, the certification path references, the
561 Attribute Authorities and CA certificate references and the revocation data references (for both the certificates
562 in the certification path and in the list of Attribute Authorities certificate).

- Time-stamp only the references.

For both ways signer, verifier or both may request, obtain and add the time-stamp to the electronic signature.

When an OCSP response is used, it is necessary to time-stamp in particular that response in the case the key from the responder would be compromised. Since the information contained in the OCSP response is user specific and time specific, an individual time-stamp is needed for every signature received. Instead of placing the time-stamp only over the certification path references and the revocation information references, which include the OCSP response, the time-stamp is placed on the digital signature (`ds:SignatureValue` element), the signature time-stamp(s) present in the XAdES-T form, the certification path references and the revocation status references (by adding a `+SigAndRefsTimeStamp` to the electronic signature). For the same cryptographic price, this will provide an integrity mechanism over the electronic signature. Any modification can be immediately detected. It should be noticed that other means of protecting/detecting the integrity of the electronic signature exist and could be used.

Despite the fact that this is the best scenario for using a time-stamp that covers both the references and the signature elements, it may also be used in scenarios where the revocation data are CRLs. However, time-stamping each electronic signature with the complete validation data references as defined above may not be efficient, particularly when the same set of CA certificates and CRL information is used to validate many signatures.

Time-stamping CA certificates will stop any attacker from issuing bogus CA certificates that could be claimed to exist before the CA key was compromised. Any bogus time-stamped CA certificates will show that the certificate was created after the legitimate CA key was compromised. In the same way, time-stamping CA CRLs, will stop any attacker from issuing bogus CA CRLs which could be claimed to exist before the CA key was compromised.

Time-stamping references to commonly used certificates and CRLs (the time-stamp token that `RefsOnlyTimeStamp` unsigned property encapsulates), can be done centrally, e.g. inside a company or by a service provider. This method reduces the amount of data the verifier has to time-stamp, for example it could reduce to just one time-stamp per day (i.e. in the case where all the signers use the same CA and the CRL applies for the whole day). As said before, the information that needs to be time-stamped is not the actual certificates and CRLs but the unambiguous references to those certificates and CRLs.

D.13 Validation data

A verifier will have to prove that the certification path was valid, at the time of the validation of the signature, up to a trust point according to the naming constraints and the certificate policy constraints from an optionally specified signature validation policy. It will be necessary to capture all the certificates from the certification path, starting with those from the signer and ending up with those of the certificate from one trusted root. Also the certificates used for validating any attribute certificate and/or time-stamp present within the electronic signature.

When dealing with long term electronic signatures, all the data used in the validation (including the certification paths of the signing certificate, and any attribute certificate and/or time-stamp present) have to be conveniently stored and time-stamped.

Several elements within a XAdES signature are entitled to contain certificate values that have been used to validate it, namely: the `ds:KeyInfo`, the `CertificateValues`, `AttrAuthoritiesCertValues` and the `CertificateValues` child of `xadesv141:TimeStampValidationData`.

The `ds:KeyInfo`, the `CertificateValues`, and the `AttrAuthoritiesCertValues` element contain the full set of certificates that have been used to validate the electronic signature, until the first archive time-stamp is added. The `CertificateValues` child of `xadesv141:TimeStampValidationData` contains certificates required for validating the time-stamp token present in the archive time-stamp.

For dealing with long term signatures, it is also needed to store and conveniently time-stamp all the revocation data used in the validation of such signatures.

When using CRLs to get revocation information, a verifier will have to make sure that he or she gets at the time of the first validation the appropriate certificate revocation information from the signer's CA. Usually this is done as soon as possible, after the grace period, to minimize the time delay between the generation and validation of the signature. This involves checking that the signer certificate serial number is not included in the CRL. The signer, the verifier or any other third party may obtain either this CRL. If obtained by the signer, then it will be conveyed to the verifier. Additional CRLs for the CA certificates in the certificate path need to also be checked by the verifier. It may be

512 convenient to archive these CRLs within an archived electronic signature for ease of subsequent validation or
513 arbitration.

514 When using OCSP to get revocation information, a verifier will have to make sure that she or he gets at the time of the
515 first validation an OCSP response that contains the status "valid". Usually this is done as soon as possible after the
516 generation of the signature, after the grace period. The signer, the verifier or any other third party may fetch this OCSP
517 response. Since OCSP responses are transient and thus are not archived by any TSP including CA, it is the
518 responsibility of every verifier to make sure that it is stored in a safe place.

519 Several elements within a XAdES signature are entitled to contain revocation data values that have been used to
520 validate it, namely: the `ds:KeyInfo`, the `RevocationValues`, `AttributeRevocationValues` and the
521 `RevocationValues` child of `xadesv141:TimeStampValidationData`.

522 The `ds:KeyInfo`, the `RevocationValues`, and the `AttributeRevocationValues` element contain the
523 full set of revocation data that have been used to validate the electronic signature, until the first archive time-stamp is
524 added. The `RevocationValues` child of `xadesv141:TimeStampValidationData` contains revocation data
525 required for validating the time-stamp token present in the archive time-stamp.

526 D.13 Time-stamps for archival

527 Advances in computing increase the probability of being able to break algorithms and compromise keys. There is
528 therefore a requirement to be able to protect electronic signatures against this possibility.

529 Over a period of time weaknesses may occur in the cryptographic algorithms used to create an electronic signature (e.g.
530 due to the time available for cryptanalysis, or improvements in cryptanalytical techniques). Before such weaknesses
531 become likely, a verifier should take extra measures to maintain the validity of the electronic signature.

532 Several techniques could be used to achieve this goal depending on the nature of the weakened cryptography. In order
533 to simplify matters, the present document specifies the Archive validation data technique, covering all the cases.

534 Archive validation data consists of the complete validation data and the complete certificate and revocation data, time-
535 stamped together with the electronic signature. The Archive validation data is necessary if the hash function and the
536 crypto algorithms that were used to create the signature are no longer secure. Also, if it cannot be assumed that the hash
537 function used by the Time-Stamping Authority is secure, then nested time-stamps of Archived Electronic Signature are
538 required.

539 The potential for Trusted Service Provider (TSP) key compromise should be significantly lower than for user keys,
540 because TSP(s) are expected to use stronger cryptography and better key protection. It can be expected that new
541 algorithms (or old ones with greater key lengths) will be used. In such a case, a sequence of time-stamps will protect
542 against forgery. Each time-stamp needs to be affixed before either the compromise of the signing key or of the cracking
543 of the algorithms used by the TSA. TSAs (Time-Stamping Authorities) should have long keys and/or a "good" or
544 different algorithm.

545 Nested time-stamps will also protect the verifier against key compromise or cracking the algorithm on the old electronic
546 signatures.

547 The process will need to be performed and iterated before the cryptographic algorithms used for generating the previous
548 time-stamp are no longer secure. Archive validation data MAY thus bear multiple embedded time-stamps.

549 D.14 The `xadesenv111:RenewedDigests` element

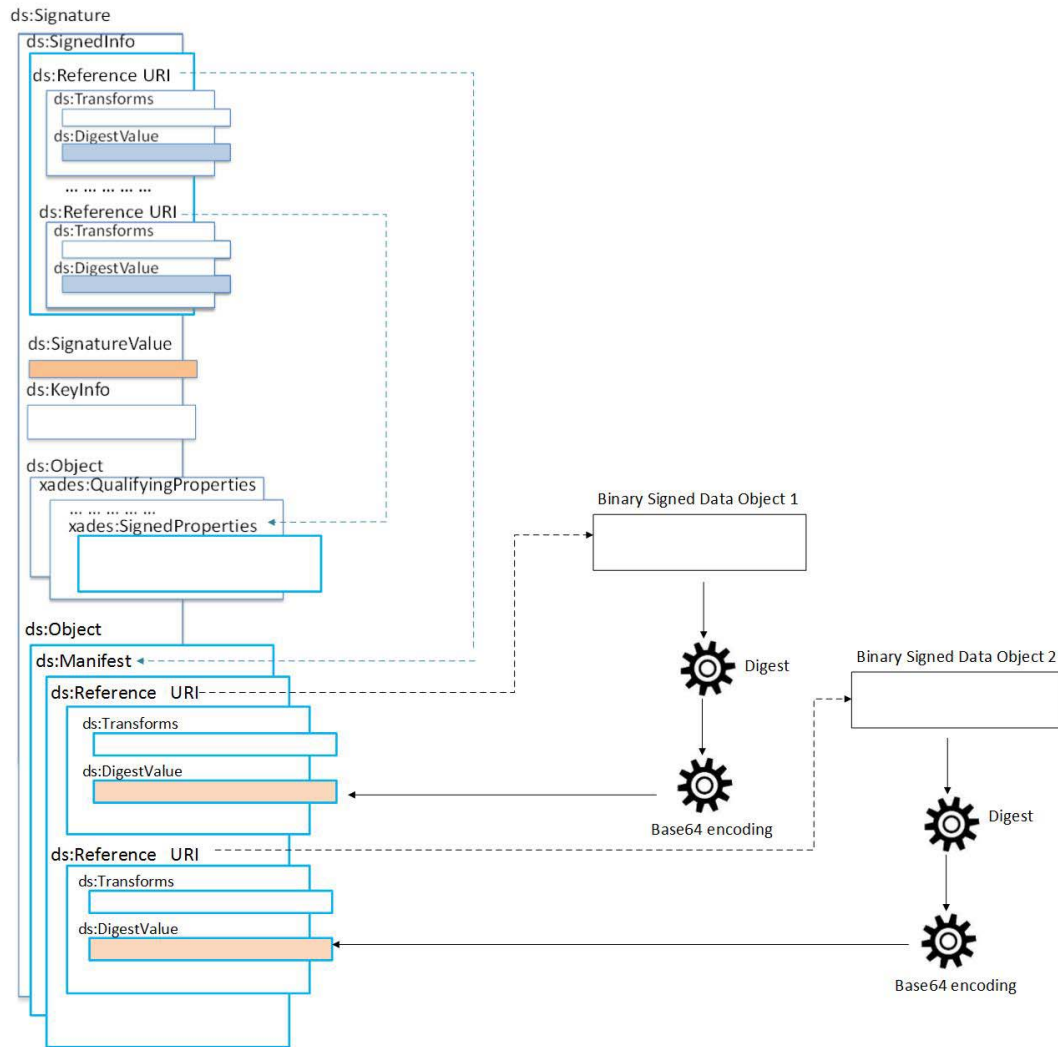
550 Figure 3 shows a XAdES-A signature that indirectly signs two detached binary data objects through one signed
551 `ds:Manifest` element.

552 The figure also shows the input to the message imprint computation (IN_{MI}) that is used for generating the time-stamp
553 token encapsulated within the `xadesv141:ArchiveTimeStamp` unsigned property, namely the concatenation of :
554 the canonicalized `xades:SignedProperties` element, the canonicalized `ds:Manifest` element, the
555 canonicalized `ds:SignedInfo` element, the canonicalized `ds:SignatureValue` element, the canonicalized
556 `ds:KeyInfo` element, the canonicalized unsigned properties that have been incorporated just before the computation
557 of the `xadesv141:ArchiveTimeStamp` in their order of appearance, and the the canonicalized `ds:Object` that
558 contains the signed `ds:Manifest`. It is worth to emphasize that the contents of the two detached data objects are not

559 part of the message imprint computation input; however their digest values, present within the `ds:Manifest`, are part
 560 of the message imprint computation input.

561

562



563

564

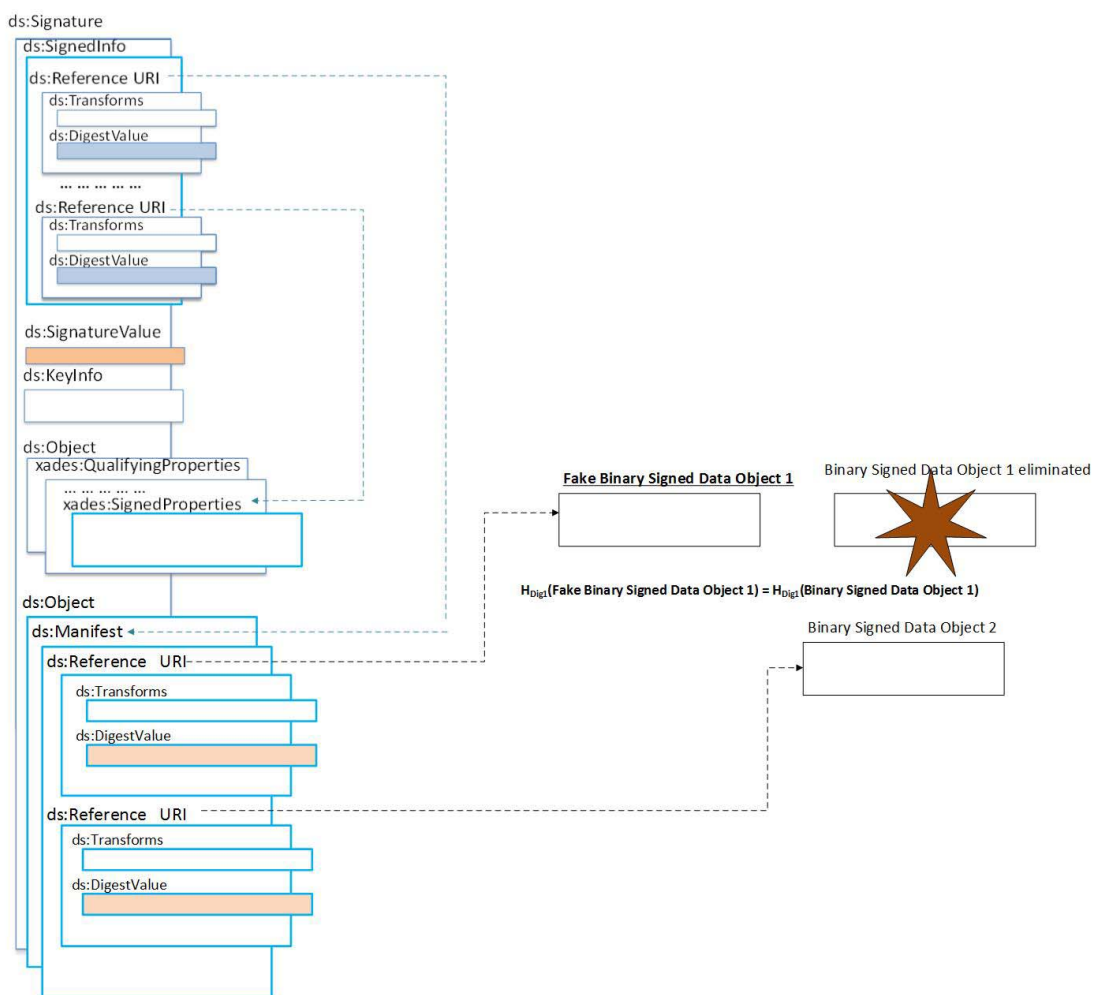
	Canonicalized SignedProperties	Canonicalized Manifest	Canonicalized SignedInfo	Canonicalized SignatureValue	Canonicalized KeyInfo	Canonicalized existing UnsignedProperties	Canonicalized Object containing Manifest
IN _{Mi} =							

Figure 3: A XAdES signature signing two detached data objects through one signed `ds:Manifest`

565

566

Figure 4 shows a potential attack to XAdES signatures built as shown in Figure 3. Such an attack will remain unnoticed if no additional measures are taken.



IN_{M1} computed by party that validates the signature

IN _{M1} =	Canonicalized SignedProperties	Canonicalized Manifest	Canonicalized SignedInfo	Canonicalized SignatureValue	Canonicalized KeyInfo	Canonicalized existing UnsignedProperties	Canonicalized Object containing Manifest

The validation of the signature would actually succeed, as all the contributions to the message imprint computation remain unchanged. However, one of the remote signed data objects has been changed!! The algorithm for building the message imprint computation of xadesv141:ArchiveTimeStamp does not counter the threat of a break in the digest algorithm used for remote data object indirectly signed by a ds:Manifest, as the original document may be changed by another one whose digest is the same as the original

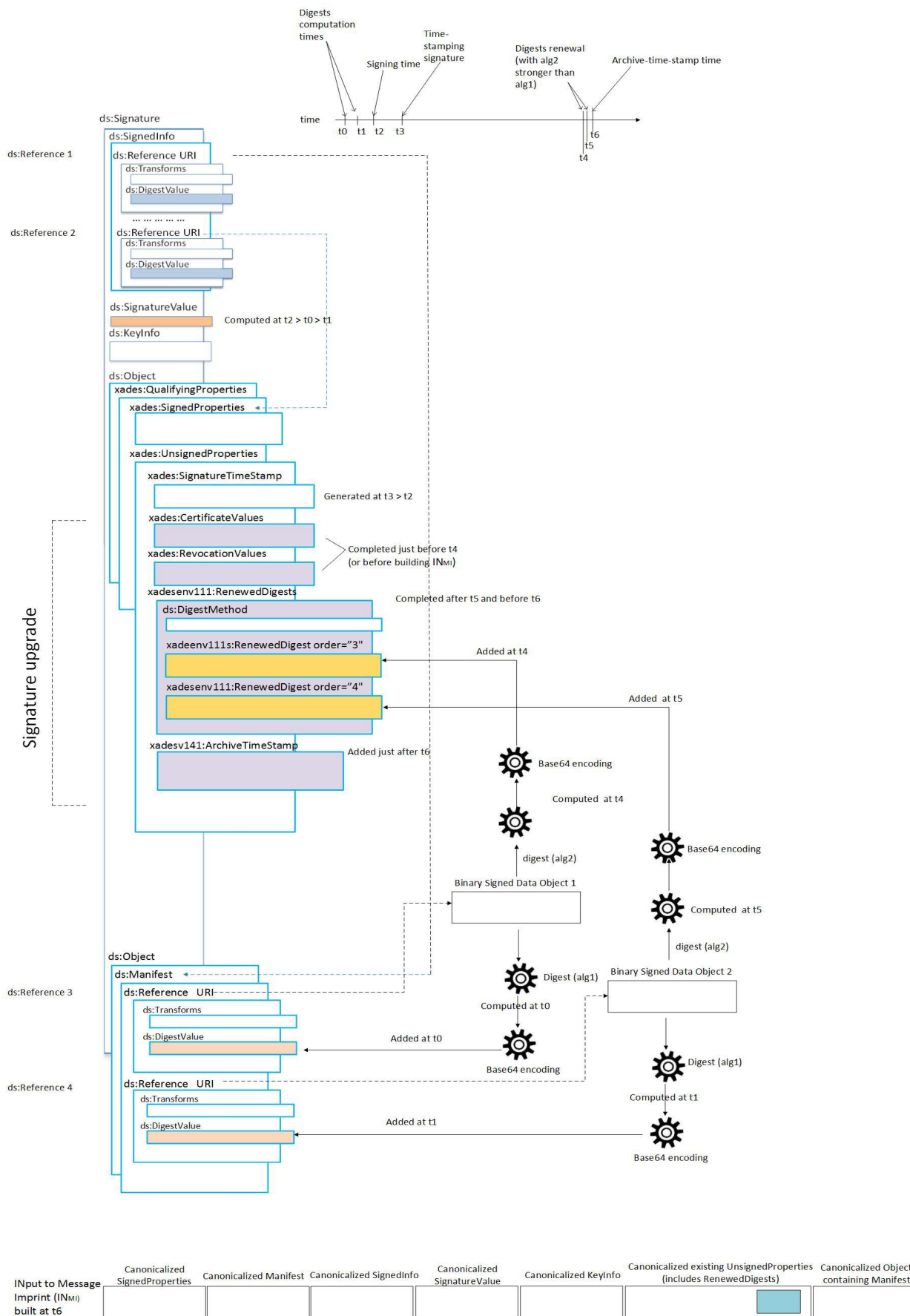
Figure 4: A successful attack: alg1 is broken and one of the data objects is substituted by another one with the same digest value

The figure shows what would happen if at a certain point in time the digest algorithm alg1 that was used for computing the digest values of the two detached data objects is broken, and if one of the aforementioned detached data objects is substituted with a fake data object whose digest value is exactly the same as the original one.

Under these circumstances, the validation of the XAdES signature would give a valid result. In first place, the check of the digest values within the signed ds:Manifest's ds:Reference children would succeed, as the digest value of the fake data object would be the same as the digest value of the original data object. In second place the validation of the xadesv141:ArchiveTimeStamp would also succeed, as the actual contents of the detached data object referenced from the signed ds:Manifest did not contribute to the computation of the message imprint.

The xadesenv111:RenewedDigests unsigned property prevents that this attack succeeds, as it is shown in figure 5 below.

581



582

583

Figure 5: Use of xadesenv111:RenewedDigests for countering the threat of figure 4

584 If the entity in charge of upgrading the XAdES signature by incorporating a `xadesv141:ArchiveTimeStamp`
585 suspects that the algorithm `alg1` is near to be broken and that some of the detached data objects referenced within the
586 signed `ds:Manifest` might be substituted by another, that entity may build the
587 `xadesenv111:RenewedDigests` unsigned property and incorporate it to the signature before computing the
588 message imprint. Under these circumstances, the input to the message imprint would be: the canonicalized
589 `xades:SignedProperties` element, the canonicalized `ds:Manifest` element, the canonicalized
590 `ds:SignedInfo` element, the canonicalized `ds:SignatureValue` element, the canonicalized `ds:KeyInfo`
591 element, the canonicalized present unsigned properties in their order of appearance, and the the canonicalized
592 `ds:Object` element that contains the signed `ds:Manifest`. It is important to emphasize two things:

- 593 1) That the `xadesenv111:RenewedDigests` unsigned property is part now of the input to the message
594 imprint and that its contents (the new digest values on the detached data objects) are secured by the time-stamp
595 token.
- 596 2) That the actual contents of the detached data objects are not part of the message imprint. This is an important
597 fact that allows that if the attack explained above succeeds, the global validation of the signature succeeds and
598 yet identify that oneo of the detached signed data objects has been substituted, as explained below.

599 If an attacker achieves to substitute one of the original detached data objects signed through the signed `ds:Manifest`
700 with another one with the same digest value computed with the broken algorithm `alg1`, then an entity validating this
701 XAdES signature will:

- 702 1) Successfully validate the time-stamp token encapsulated within the `xadesv141:ArchiveTimeStamp`, as
703 none of the elements that were concatenated for computing the input to its message imprint have been
704 changed. This ensures that whatever is covered by the time-stamp token has not been altered.
- 705 2) When checking the digest values of the `xadesenv111:RenewedDigests` unsigned property, the
706 validator will detect a failure, as the fake detached data object does not have the same digest value than the
707 original one using the algorithm `alg2`, stronger than `alg1`. This will allow the validator to conclude that the data
708 object referenced by first the `ds:Manifest`'s `ds:Reference` child element, is not the original one, and in
709 consequence, the signature on this particular object has to be considered as not valid.
- 710 3) Allow the validator still to consider the signature valid on the rest of the signed data objects if the rest of the
711 processes performed for validating the signature succeed, as the success of the archive time-stamp validation
712 ensures that nothing within the signature has changed.

713 Usage of `xadesenv111:RenewedDigests` unsigned property achieves two effects: first of all it counters the
714 threat resulting of the combination of digest algorithm break and detached data object substitution; and secondly it
715 allows to identify the substituted data object and preserve the validity of the signature for the not substituted detached
716 data objects.

717

718

719 <PAGE BREAK>

720

Annex E (informative):

721

Change History

date	Version	Information about changes
November 2013	V0.0.3	Include an extension mechanism to <code>xades:SignedSignatureProperties</code> and <code>xades:SignedDataObjectProperties</code> . Restructuring of the document to align with EN 319 122, addition of new properties: <code>xadesenv111:SigningCertificate</code> , <code>xadesenv111:SignerRole</code> , <code>xadesenv111:SignaturePolicyStorage</code> , <code>xadesenv111:RenewedDigest</code> , and <code>xadesenv111:SPDocSpecification</code> qualifier and the introduction of different conformance levels.

722

723

724

Draft

725

History

Document history		
<Version>	<Date>	<Milestone>
0.0.4	30/11/2013	Stable draft version for public comments.

726

Draft