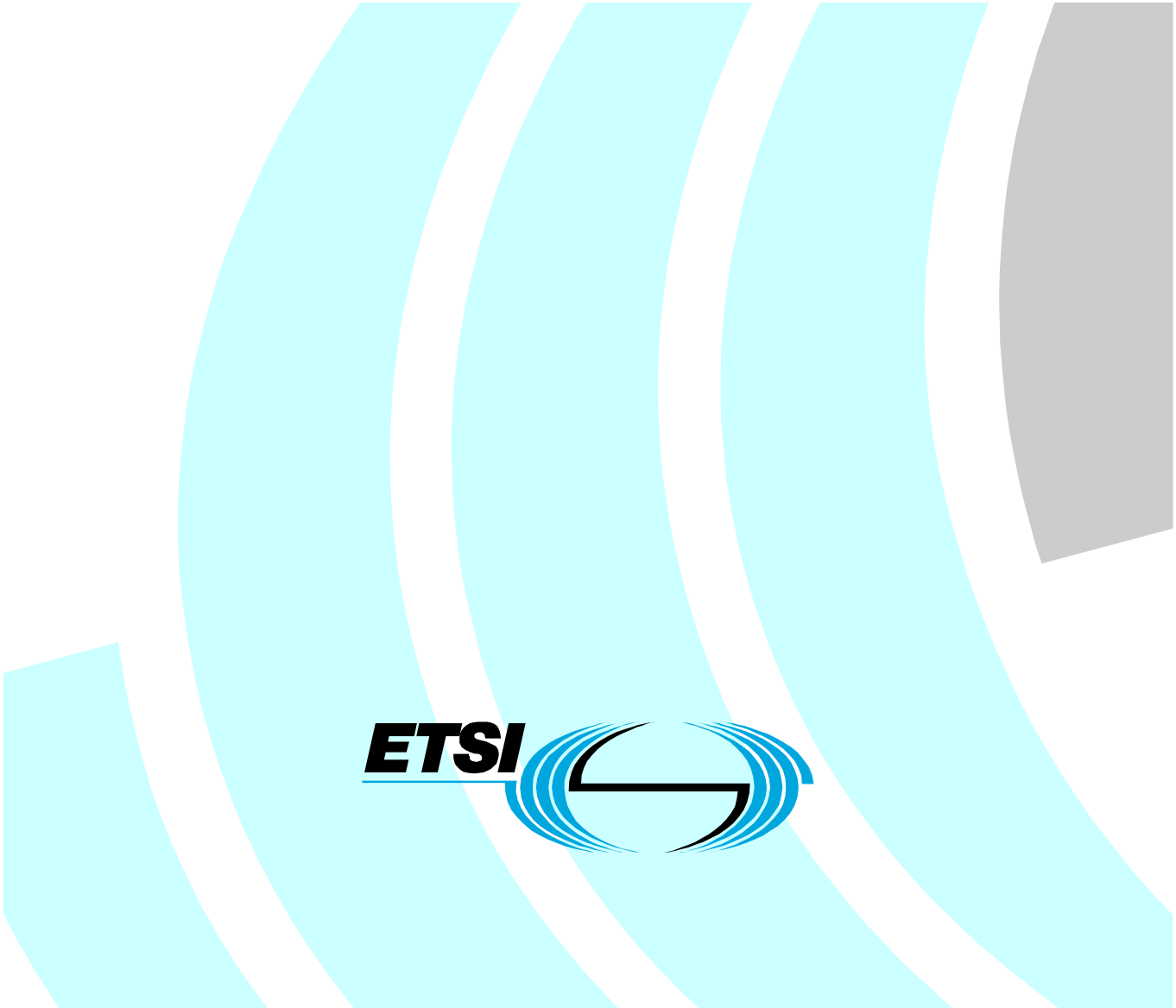
ETSI TS 101 903 V1.1.1 (2002-02)

Technical Specification

XML Advanced Electronic Signatures (XAdES)



Reference
DTS/SEC-004008

Keywords
electronic signature, security

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

Contents

Intellectual Property Rights

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

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

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Security (SEC).

Introduction

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

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

The present document is intended to cover electronic signatures for various types of transactions, including business transactions (e.g. purchase requisition, contract, and invoice applications). Thus the present document can be used for any transaction between an individual and a company, between two companies, between an individual and a governmental body, etc.

An electronic signature produced in accordance with the present document provides evidence that can be processed to get confidence that some commitment has been explicitly endorsed under a signature policy, at a given time, by a signer under an identifier, e.g. a name or a pseudonym, and optionally a role. The signature policy specifies the technical and procedural requirements on signature creation and validation in order to meet a particular business need. A given legal/contractual context may recognize a particular signature policy as meeting its requirements. For example, a specific signature policy may be recognized by court of law as meeting the requirements of the European Directive for electronic commerce.

The ETSI standard TS 101 733 [1] defines formats for advanced electronic signatures that remain valid over long periods, are compliant with the European Directive and incorporate additional useful information in common use cases (like indication of the commitment got by the signature production). Currently, it uses Abstract Syntax Notation 1 (ASN.1) and is based on the structure defined in RFC 2630 [3] (in the present document the signatures aligned with this RFC will be denoted as CMS signatures).

TS 101 733 [1]:

- Defines new ASN.1 types able to contain information for qualifying the CMS signatures so that they fulfil the aforementioned requirements.
- Specifies how this qualifying information must be incorporated to the CMS signatures.

Currently, the IETF W3C XML-Signature Working Group has developed a syntax for XML signatures: "XML-Signature Core Syntax and Processing" [5] (denoted as XMLDSIG in the present document). This syntax provides a basic functionality for digitally signing several data objects at the same time. It also provides basic means to incorporate any kind of needed qualifying information.

The present document defines XML formats for advanced electronic signatures that remain valid over long periods, are compliant with the European Directive and incorporate additional useful information in common uses cases, by:

- Proposing XML schema ([6] and [7]) definitions for new XML types able to contain the information needed to fulfil the requirement of long term validity and those ones imposed by current use cases and the European Directive. These signatures will be built on XMLDSIG by addition of this information as specified in [5], using the `ds:Object` XML element defined there (here, as for the rest of the document, `ds` has been used as the prefix denoting the namespace defined in [5]. Its value is defined in clause 4).
- Specifying the mechanisms used to produce the aforementioned addition of this qualifying information.

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

The XML advanced electronic signatures defined in the present document will be built by incorporating to the XML signatures as defined in [5] XMLDSIG one new `ds:Object` XML element containing the additional qualifying information.

Editorial conventions

As it has been anticipated in the former clause, throughout the rest of the document the term XMLDSIG will refer to XML signatures with basic functionality, i.e. to XML signatures that do not incorporate the qualifying information on the signature, the signer or the signed data object(s) specified in the present document.

Throughout the rest of the document the terms "qualifying information", "properties" or "qualifying properties" will be used to refer to the information added to the XMLDSIG to get an XML advanced electronic signature as specified in the European Directive and with long term validity.

For the present document the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in the present document are to be interpreted as described in RFC 2119 [14].

1 Scope

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

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

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

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

The present document then, specifies the use of the corresponding trusted service providers (e.g. time-stamping authorities), and the data that needs to be archived (e.g. cross certificates and revocation lists). An advanced electronic signature aligned with the present document can, in consequence, be used for arbitration in case of a dispute between the signer and verifier, which may occur at some later time, even years later.

The present document builds on the standards for Electronic Signatures defined in:

- IETF W3C: "XML-Signature Syntax and Processing" [5].
- ETSI TS 101 733: "Electronic Signature Formats" [1].
- ITU-T Recommendation X.509: "Information technology - Open Systems Interconnection - The Directory: Authentication framework" [9].
- ETSI TS 101 861: "Time stamping profile" [10].

NOTE: See clause 2 for a full set of references.

The present document, being built on the framework defined in [5] makes use of the terms defined there. Some of the definitions in [5] are repeated in the present document for the sake of completeness.

The present document:

- Shows a taxonomy of the qualifying information (properties) that have to be present in an electronic signature to remain valid over long periods, to satisfy common use cases requirements, and to be compliant with the European Directive.
- Specifies XML schema definitions for new elements able to carry or to refer to the aforementioned properties.
- Specifies two ways for incorporating the qualifying information to XMLDSIG, namely either by direct incorporation of the qualifying information or using references to such information. Both ways make use of mechanisms defined in XMLDSIG.

Clause 2 in the present document contains references to relevant documents and standards.

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

Clause 5 contains the namespace specification for the XML schema definitions appearing in the present document.

Clause 6 describes how the qualifying information is added to XMLDSIG.

Clause 7 contains the details (including schema definitions) of the elements where the qualifying information is included.

Annex A contains definitions for relevant concepts used throughout the present document.

Annex B contains the whole set of schema definitions for the elements defined in the present document.

Annex C contains the non normative DTD corresponding to the aforementioned schema.

Annex D shows examples of how to incorporate qualifying information leading to the XML Advanced Electronic Signatures.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] ETSI TS 101 733: "Electronic Signature Formats".

[2] Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures.

[3] RFC 2630: "Cryptographic Message Syntax".

[4] RFC 2459: "Internet X.509 Public Key Infrastructure Certificate and CRL Profile".

[5] W3C 08-2001 (W3C/IETF Proposed Recommendation, August 2001): "XML-Signature Syntax and Processing".

[6] W3C Recommendation: "XML Schema Part 1: Structures".

[7] W3C Recommendation: "XML Schema Part 2: Datatypes".

[8] RFC 2634: "Enhanced Security Services for S/MIME".

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

[10] ETSI TS 101 861: "Time stamping profile".

[11] RFC 2396: "Uniform Resource Identifiers (URI): Generic Syntax".

[12] W3C: "Extensible Markup Language (XML) 1.0".

[13] RFC 2560: "X.509 Internet Public Key Infrastructure Online Certificate Status Protocolol - OCSP".

[14] RFC 2119: "Key words for use in RFCs to Indicate Requirement Levels".

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

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

[17] W3C 12-2000 (W3C Candidate Recommendation, December 2000): "The Platform for Privacy Preferences 1.0 (P3P1.0) Specification".

3 Abbreviations

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

ARL	Authority Revocation List
CA	Certification Authority
CRL	Certificate Revocation List
CMS	Cryptographic Message Syntax
DTD	Document Type Definition
OCSP	Online Certificate Status Protocol
OID	Object IDentifier
SP	Signature Policy

TSA	Time-Stamping Authorities
TSP	Trusted Service Providers
URI	Uniform Resource Identifier
URN	Uniform Resource Name
XAdES	XML Advanced Electronic Signature
XAdES-C	XAdES with Complete validation data
XAdES-T	XAdES with Time-Stamp
XML	extensible Markup Language
XMLDSIG	XML-Signature Syntax and Processing

4 Overview

4.1 Major Parties

The following are the major parties involved in a business transaction supported by electronic signatures as defined in the present document:

- the Signer;
- the Verifier;
- Trusted Service Providers (TSP);
- the Arbitrator.

The **Signer** is the entity that creates the electronic signature. When the signer digitally signs over data object(s) (see definition) using the prescribed format, this represents a commitment on behalf of the signing entity to the data object(s) being signed.

The **Verifier** is the entity that verifies the electronic signature. It may be a single entity or multiple entities.

The **Trusted Service Providers** (TSPs) are one or more entities that help to build trust relationships between the signer and verifier. They support the signer and verifier by means of supporting services including user certificates, cross-certificates, time-stamping tokens, CRLs, ARLs, OCSP responses. The following TSPs are used to support the functions defined in the present document:

- Certification Authorities;
- Registration Authorities;
- Repository Authorities (e.g. a directory);
- Time-Stamping Authorities;
- Signature Policy Issuers.
- Attribute Authorities.

Certification Authorities (CA) provide users with public key certificates.

Registration Authorities allow the identification and registration of entities before a CA generates certificates.

Repository Authorities publish CRLs issued by CAs, signature policies issued by signature policy issuers and optionally public key certificates.

Time-Stamping Authorities (TSA) attest that some data object was formed before a given trusted time.

Signature Policy Issuers define the technical and procedural requirements for electronic signature creation and validation, in order to meet a particular business need.

Attributes Authorities provide users with attributes linked to public key certificates.

An **Arbitrator** is an entity that arbitrates in disputes between a signer and a verifier.

4.2 Electronic signatures and validation data

Validation of an electronic signature in accordance with the present document requires:

- A XML advanced electronic signature built on the format defined in [5] with the incorporation of additional qualifying information. This XML advanced electronic signature will include:
  - references to the **signed data object(s)** (as specified in [5]);
  - **signed properties** (provided by the signer);
  - the **signature** itself as defined in [5] (see definitions).
- Validation data, which is the additional data needed to validate the electronic signature; this includes:
  - certificates;
  - revocation status information;
  - time-stamp tokens from Time-Stamping Authorities (TSAs).

Signed data object(s) is the user's data that is signed.

Signed properties include any additional information that shall be signed by the signer to conform to the signature policy or the present document (e.g. signing time).

The **Validation Data** may be **collected by the signer and/or the verifier** and shall meet the requirements of the signature policy. Additional data includes CA certificates as well as revocation status information in the form of certificate revocation lists (CRLs) or certificate status information provided by an on-line service. Additional data also includes time-stamps and other time related data used to provide evidence of the timing of certain events. It is required, as a minimum, that either the signer or verifier obtains a time-stamp over the signer's signature or a record must be maintained and cannot be undetectable modified, of the electronic signature and the time when the signature was first validated.

4.3 XML Advanced Electronic Signature Data Structures

The present document defines different forms of electronic signatures, each one satisfying requirements that will be shown in the corresponding clauses.

The current clause presents the first three forms: the XML Advanced Electronic Signature (XAdES), the XAdES with Time-Stamp (XAdES-T) and the XAdES with Complete VALIDATION DATA (XAdES-C). Clause 4.4 introduces extended forms to the XAdES (XAdES-X and XAdES-X-L) to meet additional requirements. Finally, clause 4.5 presents the format for archiving signatures in a way that they are protected if the cryptographic data become weak (XAdES-A).

The first three forms are the following ones:

The XML Advanced Electronic Signature (XAdES). Its format is the one defined in [5] with the **addition of signed properties** (`SigningTime`, `SigningCertificate`, `SignaturePolicyIdentifier`, `SignatureProductionPlace`, `SignerRole`, `AllDataObjectsTimeStamp`, `IndividualDataObjectsTimeStamp`, `DataObjectFormat` and `CommitmentTypeIndication`) and **unsigned properties** (`CounterSignature`).

- The XML Advanced Electronic Signature with Time-Stamp (XAdES-T), **which adds a time-stamp** to the XAdES, to take initial steps towards providing long term validity. This form or a time record should be created close to the time that the XAdES was produced to provide protection against repudiation.

- The XML Advanced Electronic Signature with Complete validation data (XAdES -C), which adds to the XAdES-T the **references to the set of data supporting the validity of the electronic signature** (i.e. the remaining references to the certification path and its associated revocation status information). Note that this does **not** contain the actual data of the certification path and its associated revocation status information, which is far more voluminous.

The XAdES satisfies the legal requirements for advanced electronic signatures as defined in the European Directive on electronic signatures. It provides basic authentication and integrity protection and can be created without accessing on-line (time-stamping) services. However, without the addition of a time-stamp or a secure time record the electronic signature does not protect against the threat that the signer later denies having created the electronic signature (i.e. does not provide non-repudiation of its existence).

The XAdES-T timestamp should be created close to the time that XAdES was created to provide protection against repudiation. At this time all the data needed to complete the validation may not be available but what information is readily available may be used to carry out some of the initial checks. For example, only part of the revocation information may be available for verification at that point in time.

Support for XAdES-C by the verifier is mandated as soon as there is a need for a subsequent verification.

The signer shall provide at least the XAdES form, but in some cases may decide to provide the XAdES-T form and in the extreme case could provide the XAdES-C form. If the signer does not provide XAdES-T, the verifier shall either create the XAdES-T on first receipt of an electronic signature or shall keep a secure record of the current time with the XAdES. Either of these two approaches provide independent evidence of the existence of the signature at the time it was first verified which should be near the time it was created, and so protects against later repudiation of the existence of the signature. If the signer does not provide XAdES-C the verifier shall create the XAdES-C when the complete set of revocation and other validation data is available. Generally, the XAdES-C form cannot be created at the same time as the XAdES, as it is necessary to allow time for any revocation information to be captured. Also, if a certificate is found to be temporarily suspended, it will be necessary to wait until the end of the suspension period.

The signer should only create the XAdES-C in situations where it was prepared to wait for a sufficient length of time after creating the XAdES form before dispatching the XAdES-C. This, however, has the advantage that the verifier can be presented with the complete set of data supporting the validity of the XAdES.

An XML Advanced Electronic Signature XAdES is illustrated in figure 1.

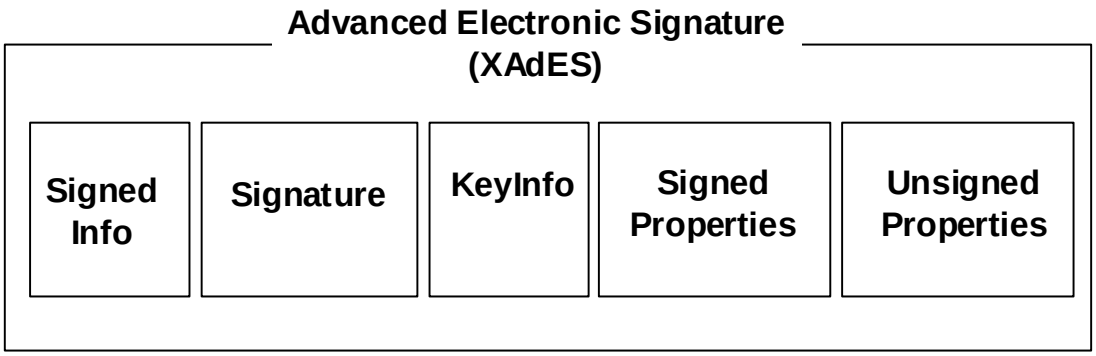
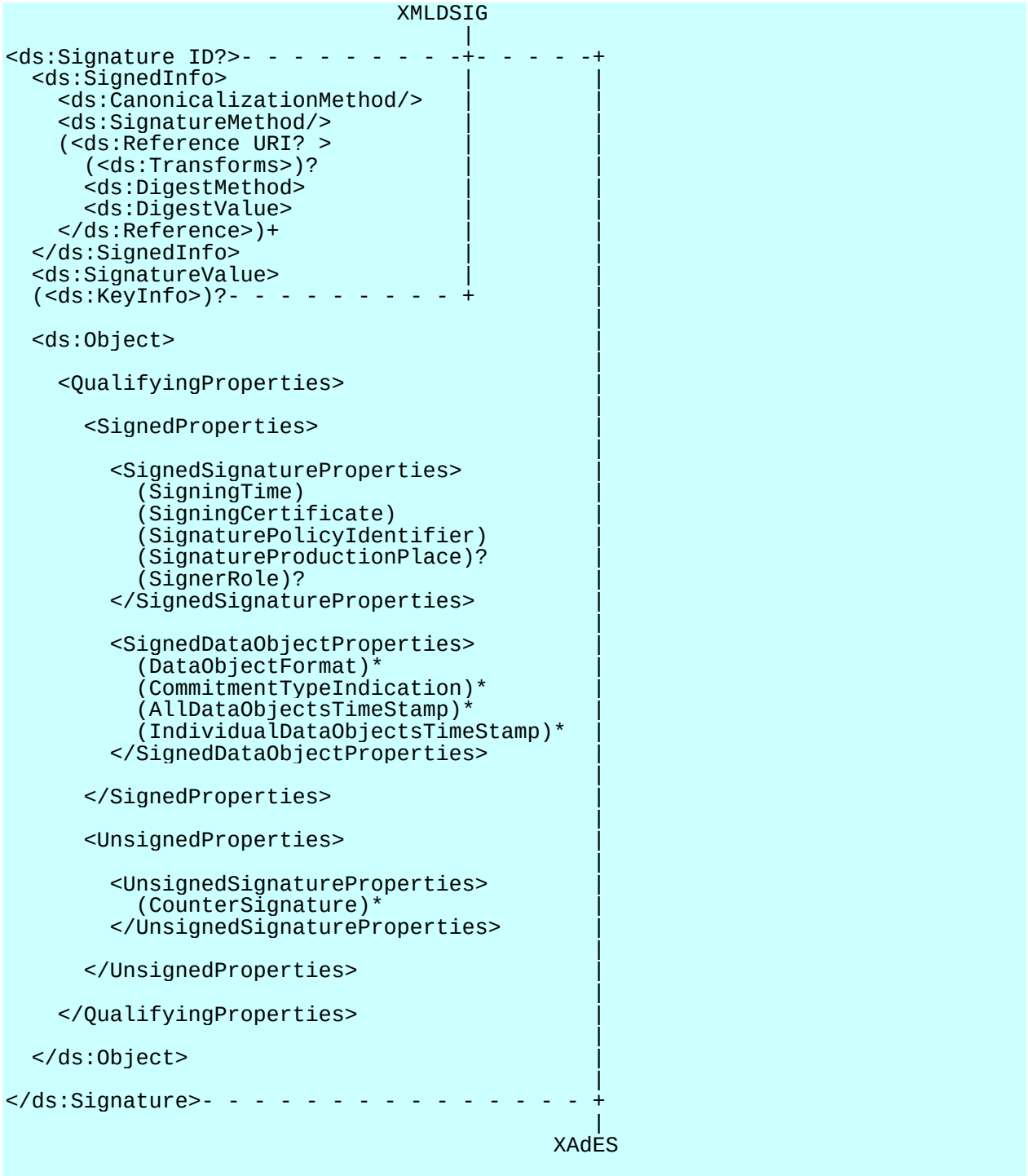


Figure 1: Illustration of an XAdES

In figure 1, `SignedInfo`, `Signature` and `KeyInfo` are as defined in XMLDSIG [5].

Below follows the structure of the XAdES built **by direct incorporation** of the qualifying information in the corresponding new XML elements to the XMLDSIG (see clause 6.3 for further details). In the example "?" denotes zero or one occurrence; "+" denotes one or more occurrences; and "*" denotes zero or more occurrences.

The XML schema definition in clause 5 defines the prefix "`ds`" for all the XML elements already defined in XMLDSIG, and states that the default namespace is the one defined for the present document. In consequence, in the examples of this clause, the elements already defined in XMLDSIG appear with the prefix "`ds`", whereas the new XML elements defined in the present document appear without prefix.



Readers must take into account that the XAdES forms build up on the XMLDSIG by adding new XML elements containing qualifying information within the shown XMLDSIG `ds:Object` element, according to the rules defined in the present document. This `ds:Object` element will act as a bag for the whole set of qualifying properties defined in the present document, conveniently grouped.

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

Detailed explanation of the purposes of each property will be given throughout clause 7.

An XML Advanced Electronic Signature (XAdES), with the additional validation data forming the XAdES-T and XAdES-C is illustrated in figure 2.

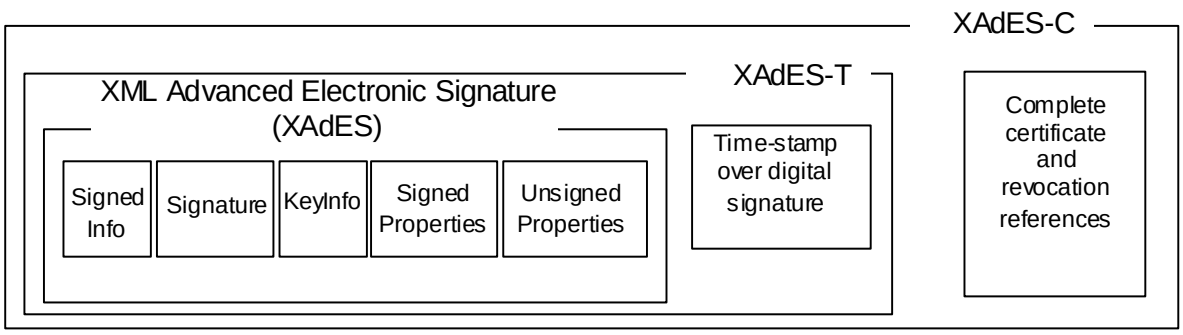


Figure 2: Illustration of an XAdES, XAdES-T and XAdES-C

Below follows the structure of XAdES-T signature.

Below follows the structure for XAdES-C.

4.3.1 Contents

4.3.1.1 Contents of XAdES

As it has been stated, a XAdES signature will build on XMLDSIG by incorporation of one `ds:Object` that will be the bag for the whole set of qualifying properties. Some of them will be signed (signed qualifying information grouped within one new element, `SignedProperties`, see clause 6.2.1) and others will not be signed (unsigned qualifying information, grouped within the `UnsignedProperties` element, see clause 6.2.2).

In a XAdES the signature SHALL be applied in the usual way of XMLDSIG over the data object(s) to be signed **and on the whole set of signed properties** (`SignedProperties` element). The mandatory information in the `SignedProperties` element is:

- an unambiguous **reference to the signer's certificate**, e.g. the certificate itself or a reference to it together with a hash value of the certificate. This 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 verification data. Finally, it is also important in case the issuing key of the CA providing the certificate would be compromised (clause 7.2.2);

- an **unambiguous way allowing the identification of the signature policy** under which the electronic signature has been produced (clause 7.2.3). This will ensure that the verifier will be able to use the same signature policy during the verification process. A signature policy is needed 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 a different signature policy was implied by the signer.

- the **signing time**, specifying the time at which the signer claims to have performed the signing process (clause 7.2.1);

In addition, the signature **can** also cover other **signed properties** containing the following information:

- the **data object(s) format(s)** that identifies the format of a signed data object (when electronic signatures are not exchanged in a restricted context) to enable the verifier to be presented or use it (text, sound or video) in exactly the same way as intended by the signer (clause 7.2.5);

- the **commitment type(s)** undertaken by the signer in signing (a) signed data object(s) in the context of the selected signature policy (when an explicit commitment is being used); This will be required where a Signature Policy specifies more than a single commitment type, each of which might have different legal interpretations of the intent of the signature (e.g. proof of origin, proof of receipt, proof of creation …) (clause 7.2.6);

- the **claimed or certified role** assumed by the signer in creating the signature (clause 7.2.8);

- the **purported place** where the signer claims to have produced the signature (clause 7.2.7);

- one or more **time-stamps on data objects to be signed** (clauses 7.2.9 and 7.2.10).

4.3.1.2 Contents of XAdES-T

The signer or the verifier can build an XAdES-T by adding to the existent XAdES (as a child of `UnsignedProperties` element), an XML element (clause 7.1.3) encapsulating a time-stamp on the XMLDSIG digital signature value, generated by a TSA to prove that the electronic signature was performed before that time (clause 7.3.1).

4.3.1.3 Contents of XAdES-C

The signer or the verifier of an electronic signature can create the XAdES-C form by **incorporating to the XAdES-T references to the complete set of data supporting its validity** (certificate path, certificate revocation lists, OCSP responses, etc). The signer or the verifier will create this form by incorporating to the XAdES-T these references, within an XML element whose definition will be given in the present document (clauses 7.4.1 and 7.4.2). This element will be added as a child of the `UnsignedProperties` element.

4.4 Extended forms of validation data

The complete validation data (XAdES-C) described above may be extended to form an XAdES with eXtended validation data (XAdES-X) to meet following additional requirements.

- Firstly, if there is a risk that any **keys** used in the certificate chain or in the revocation status information may be **compromised**. The case of a broken algorithm is different and is addressed later on in the archived form of an electronic signature. It is necessary to additionally time-stamp all the certification path references and revocation status references, contained in the XAdES-C (see clause 7.5.2). Alternatively, the time-stamp can be applied to the digital signature (`ds:Signature` element), the time-stamp(s) present in the XAdES-T form and the aforementioned references (see clause 7.5.1).

- Secondly, when the certification path data and revocation status data is not stored for the long term elsewhere, then there is a need to add them to the signature (XAdES-X-L)

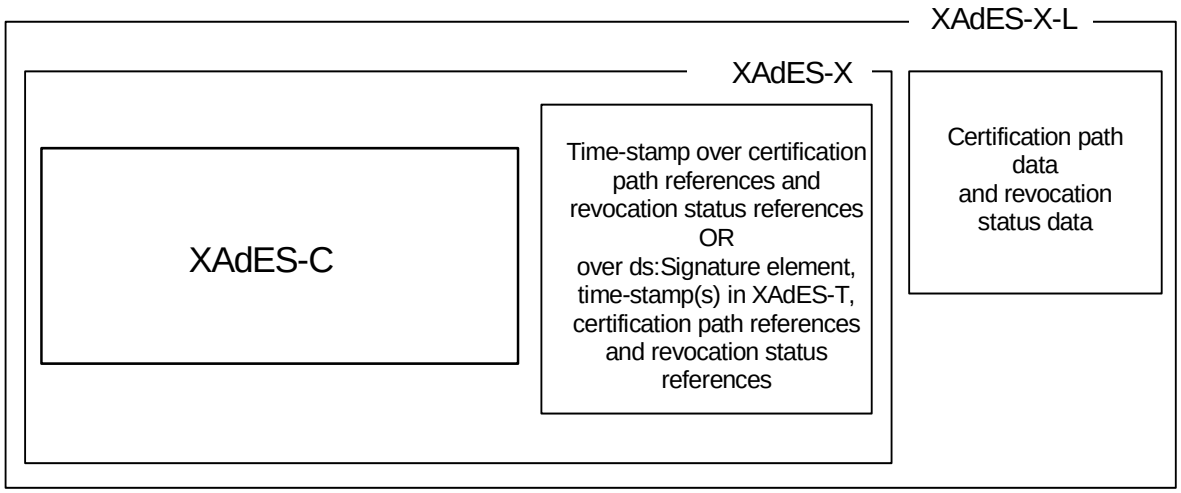


Figure 3: Illustration of an XAdES-X and XAdES-X-L

Note it may be possible to omit the time-stamp over certification path references and revocation status references while still adding the Certification path data and revocation status data.

The XAdES-X validation data is created by adding to a previously generated XAdES-C a time-stamp over the references to the complete set of data supporting its validity or over the sequence formed by `ds:SignatureValue` element, the previous time-stamp(s) present in the XAdES-T form and the aforementioned references. Again, this new form will be achieved by adding an XML element conveniently encapsulating this time-stamp (see clauses 7.5.1 and 7.5.2). This element will be added as a child of the `UnsignedProperties` element.

The XAdES-X-L will be produced by incorporating the certificate path and revocation information (CRLs or OCSP responses) conveniently encapsulated by XML elements (see clauses 7.6.1 and 7.6.2). These elements will be added as children of the `UnsignedProperties` element.

Below follows the XAdES-X structure.

The structure for XAdES-X-L is shown below.

4.5 Archive validation data

Before the algorithms, keys and other cryptographic data used at the time the XAdES-C was built become weak and the cryptographic functions become vulnerable, the XAdES-X-L should be time-stamped. If possible this should use stronger algorithms (or longer key lengths) than in the original time-stamps. This additional data and time-stamp is called Archive Validation Data (XAdES-A). The time-stamping process may be repeated every time the protection used to time-stamp a previous XAdES-A become weak. A XAdES-A may thus bear multiple embedded time-stamps).

Support for XAdES-A is optional.

An example of an XML Advanced Electronic Signature (XAdES), with the additional validation data for the XAdES-C and XAdES-X-L time-stamped forming the XAdES-A is illustrated in figure 4.

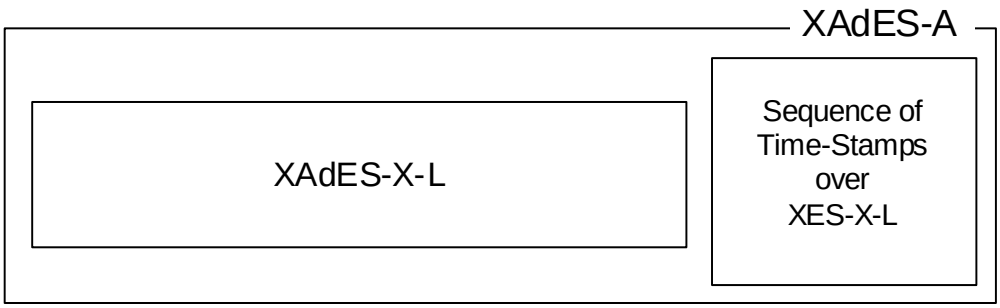


Figure 4: Illustration of XAdES-A

Below follows the structure of XAdES-A.


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)
(SigningCertificate)
(SignaturePolicyIdentifier)
(SignatureProductionPlace)?
(SignerRole)?
</SignedSignatureProperties>
<SignedDataObjectProperties>
(DataObjectFormat)*
(CommitmentTypeIndication)*
(AllDataObjectsTimeStamp)*
(IndividualDataObjectsTimeStamp)*
</SignedDataObjectPropertiesSigned>
</SignedProperties>
<UnsignedProperties>
</UnsignedSignatureProperties>
(CounterSignature)*- - - - - - - - +
(SignatureTimeStamp)+- - - - - - - - +
(CompleteCertificateRefs)
(CompleteRevocationRefs)- - - - - - -+
((SigAndRefsTimeStamp)*
(RefsOnlyTimeStamp)*)- - - - - - - - - +
(CertificatesValues)
(RevocationValues)- - - - - - - - - - - -+
(ArchiveTimeStamp)+
</UnsignedSignatureProperties>- - - -+-+-+-+-+
</UnsignedProperties>
</QualifyingProperties>
</ds:Object>
</ds:Signature>- - - - - - - - - - - - - - - +-+-+-+-+-+
XAdES
XAdES-T
XAdES-C
XAdES-X
XAdES-X-L
XAdES-A

This form will be produced by adding to the XAdES-X-L XML elements containing time-stamps conveniently encapsulated. The time-stamps will be computed over data within the previous structure: XAdES-X-L for the first one, XAdES-X-L plus other time-stamps for the following ones (see clause 7.7.1). These elements will be added as children of the `UnsignedProperties` elements.

5 XML namespace for the present document

The XML namespace URI that must be used by implementations of the present document:

http://uri.etsi.org/01903/v1.1.1#

The following namespace declarations apply for the XML Schema definitions throughout the present document:

```
<?xml version="1.0"?>
<schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://uri.etsi.org/01903/v1.1.1#"
  targetNamespace="http://uri.etsi.org/01903/v1.1.1#"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  elementFormDefault="qualified"
>
```

6 Syntax overview

This clause introduces the syntax for adding qualifying information to an XML signature.

Clause 6.1 lists a set of technical criteria that has been taken into account for this syntax proposal.

Clause 6.2 specifies an XML element that acts as a container for the qualifying information. Additionally it describes the connection between the XML signature and this container element.

Clause 6.3 shows two ways of incorporating such qualifying information to XMLDSIG.

6.1 Technical criteria

The following considerations have been taken into account for the syntax specification for qualifying information on XML signatures.

- The present document specifies how to add qualifying information to an XML signature such that it satisfies both the requirements for an Advanced Electronic Signature according to the European Directive On Electronic Signatures and for remaining valid over long period of time. TS 101 733 [1] identifies all the required information to be added in order to satisfy those requirements. Additionally it defines appropriate data structures for those qualifying properties using ASN.1, that fit for CMS [3] style electronic signatures. The aim of the present document is to specify similar XML qualifying properties that carry such qualifying information and are used to amend XMLDSIG.

- The new XML qualifying properties should not be the result of a stubborn translation process from ASN.1 to XML. This would mean neglecting syntactic differences between CMS [3] and XMLDSIG such as the possible number of signers and multiple signed data objects covered by a single signature, as well as ignoring powerful features of the XML environment such as linking information by using Uniform Resource Identifiers (URI).

- XML Schema [7] has been chosen as the normative language for defining the new XML structures in the present document rather than the DTD vocabulary defined in XML 1.0 [12], since it is namespace aware, allows reuse of existing structures and allows a stricter definition of the allowed contents. However, a DTD of the new XML structures is provided as informative annex of the present document.

- XML structures that have been defined in related XML standards such as XML Schema [7] and XML-Signature Syntax and Processing [5] have been reused where appropriate.

6.2 The QualifyingProperties element

The QualifyingProperties element acts as a container element for all the qualifying information that should be added to an XML signature. The element has the following structure:

```
<xsd:element name="QualifyingProperties" type="QualifyingPropertiesType"/>
<xsd:complexType name="QualifyingPropertiesType">
  <xsd:sequence>
    <xsd:element name="SignedProperties" type="SignedPropertiesType"
      minOccurs="0"/>
    <xsd:element name="UnsignedProperties" type="UnsignedPropertiesType"
      minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="Target" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The qualifying properties are split into properties that are cryptographically bound to (i.e. signed by) the XML signature (SignedProperties), and properties that are not cryptographically bound to the XML signature (UnsignedProperties). The SignedProperties must be covered by a Reference element of the XML signature. Alignment with the present document mandates that one SignedProperties element MUST exist.

The mandatory Target attribute refers to the XML signature with which the qualifying properties are associated.

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

6.2.1 The SignedProperties element

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

Alignment with the present document mandates that an element SignedSignatureProperties MUST appear.

Below follows the schema definition for SignedProperties element.

```
<xsd:element name="SignedProperties" type="SignedPropertiesType" />

<xsd:complexType name="SignedPropertiesType">
  <xsd:sequence>
    <xsd:element name="SignedSignatureProperties"
      type="SignedSignaturePropertiesType"/>
      <xsd:element name="SignedDataObjectProperties"
        type="SignedDataObjectPropertiesType" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The SignedProperties element MUST contain properties that qualify the XMLDSIG signature itself or the signer. They are included as content of the SignedSignatureProperties element.

The SignedProperties element MAY also contain properties that qualify some of the signed data objects. These properties appear as content of the SignedDataObjectProperties element.

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

6.2.2 The UnsignedProperties element

The UnsignedProperties element contains a number of properties that are not signed by the XMLDSIG signature.

```
<xsd:element name="UnsignedProperties" type="UnsignedPropertiesType" />

  <xsd:complexType name="UnsignedPropertiesType">
    <xsd:sequence>
      <xsd:element name="UnsignedSignatureProperties"
        type="UnsignedSignaturePropertiesType" minOccurs="0"/>
        <xsd:element name="UnsignedDataObjectProperties"
          type="UnsignedDataObjectPropertiesType" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
  </xsd:complexType>
```

The UnsignedProperties element MAY contain properties that qualify XML signature itself or the signer. They are included as content of the UnsignedSignatureProperties element.

The UnsignedProperties element MAY also contain properties that qualify some of the signed data objects. These properties appear as content of the UnsignedDataObjectProperties element.

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

6.2.3 The SignedSignatureProperties element

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

```
<xsd:element name="SignedSignatureProperties"
  type="SignedSignaturePropertiesType" />

<xsd:complexType name="SignedSignaturePropertiesType">
  <xsd:sequence>
    <xsd:element name="SigningTime" type="xsd:dateTime"/>
    <xsd:element name="SigningCertificate" type="CertIDListType"/>
    <xsd:element name="SignaturePolicyIdentifer"
     type="SignaturePolicyIdentifierType"/>
    <xsd:element name="SignatureProductionPlace"
type="SignatureProductionPlaceType"
     minOccurs="0"/>
    <xsd:element name="SignerRole" type="SignerRoleType" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

The qualifying property SigningTime is described in detail in clause 7.2.1, SigningCertificate in clause 7.2.2, SignaturePolicyIdentifier in clause 7.2.3, SignatureProductionPlace in clause 7.2.7, and SignerRole in clause 7.2.8.

6.2.4 The SignedDataObjectProperties element

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

```
<xsd:element name="SignedDataObjectProperties"
  type="SignedDataObjectPropertiesType"/>

<xsd:complexType name="SignedDataObjectPropertiesType">
  <xsd:sequence>
    <xsd:element name="DataObjectFormat" type="DataObjectFormatType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="CommitmentTypeIndication"
      type="CommitmentTypeIndicationType" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="AllDataObjectsTimeStamp" type="TimeStampType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="IndividualDataObjectsTimeStamp" type="TimeStampType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

The qualifying property `AllDataObjectsTimeStamp` is described in detail in clause 7.2.9, `IndividualDataObjectsTimeStamp` in clause 7.2.10, `DataObjectFormat` in clause 7.2.5, and `CommitmentTypeIndication` in clause 7.2.6.

All these properties qualify the signed data object after all the required transforms have been made.

6.2.5 The UnsignedSignatureProperties element

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.

```
<xsd:element name="UnsignedSignatureProperties"
  type="UnsignedSignaturePropertiesType"/>

<xsd:complexType name="UnsignedSignaturePropertiesType">
  <xsd:sequence>
    <xsd:element name="CounterSignature" type="CounterSignatureType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="SignatureTimeStamp" type="TimeStampType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="CompleteCertificateRefs"
      type="CompleteCertificateRefsType" minOccurs="0"/>
    <xsd:element name="CompleteRevocationRefs"
      type="CompleteRevocationRefsType" minOccurs="0"/>
    <xsd:choice>
      <xsd:element name="SigAndRefsTimeStamp" type="TimeStampType"
        minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="RefsOnlyTimeStamp" type="TimeStampType"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:choice>
    <xsd:element name="CertificateValues" type="CertificateValuesType"
      minOccurs="0"/>
    <xsd:element name="RevocationValues" type="RevocationValuesType"
      minOccurs="0"/>
    <xsd:element name="ArchiveTimeStamp" type="TimeStampType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  </xsd:complexType>
```

The qualifying property `CounterSignature` is described in detail in clause 7.2.4, `SignatureTimeStamp` in clause 7.3.1, `CompleteCertificateRefs` in clause 7.4.1, `CompleteRevocationRefs` in clause 7.4.2, `SigAndRefsTimeStamp` in clause 7.5.1, `RefsOnlyTimeStamp` in clause 7.5.2, `CertificateValues` in clause 7.6.1, `RevocationValues` in clause 7.6.2, and `ArchiveTimeStamp` in clause 7.7.1.

6.2.6 The UnsignedDataObjectProperties element

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

```
<xsd:element name="UnsignedDataObjectProperties"
  type="UnsignedDataObjectPropertiesType" />

<xsd:complexType name="UnsignedDataObjectPropertiesType">
  <xsd:sequence>
    <xsd:element name="UnsignedDataObjectProperty" type="AnyType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

TS 101733 [1] does not specify the usage of any unsigned property qualifying the signed data object. The present document, however, incorporates this element for the shake of completeness and to cope with potential future needs for inclusion of such kind of properties. The schema definition leaves open the definition of the contents of this type. The type AnyType is defined in clause 7.1.1.

6.3 Incorporating qualifying properties into an XML signature

The present document utilizes the `ds:Object` auxiliary element from XMLDSIG [5]. It MUST be used to incorporate the qualifying properties into the XMLDSIG signature. In principle, two different means are provided for this incorporation:

- Direct incorporation means that a `QualifyingProperties` element is put as a child of the `ds:Object`.
- Indirect incorporation means that a `QualifyingPropertiesReference` element is put as a child of the `ds:Object`. This element contains information about a `QualifyingProperties` element that is stored in place different from the signature (see clause 6.3.2).

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

- All instances of the `QualifyingProperties` and the `QualifyingPropertiesReference` element MUST occur within a single `ds:Object` element.
- At most one instance of the `QualifyingProperties` element may occur within this single `ds:Object` element.
- All signed properties must occur within a single QualifyingProperties element. This element can either be a child of the `ds:Object` element (direct incorporation), or it can be referenced by a `QualifyingPropertiesReference` element. See clause 6.3.1 for information how to sign properties.
- Zero or more instances of the `QualifyingPropertiesReference` element may occur within the single `ds:Object` element.

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

6.3.1 Signing properties

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

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

Additionally, the present document MANDATES the use of the `Type` attribute of this particular `ds:Reference` element, with its value set to

`http://uri.etsi.org/01903/v1.1.1#SignedProperties`

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

If the `QualifyingProperties` element containing the `SignedProperties` element is stored in a place different from the signature (indirect incorporation), the result of processing the URI and transforms in this `ds:Reference` element must be the same as the result of processing the URI and transforms in the `QualifyingPropertiesReference` element pointing to the aforementioned `QualifyingProperties` element.

6.3.2 The `QualifyingPropertiesReference` element

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

```
<xsd:element name="QualifyingPropertiesReference"
  type="QualifyingPropertiesReferenceType"/>

<xsd:complexType name="QualifyingPropertiesReferenceType">
  <xsd:sequence>
    <xsd:element name="Transforms" type="ds:TransformsType" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="URI" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The mandatory `URI` attribute provides an identifier for the location of the `QualifyingProperties` element. This could be for instance a URL to a web site where the information can be retrieved, or a name that the participating applications can use to identify a particular `QualifyingProperties` element.

The optional `ds:Transforms` element can be used to specify a chain of tranformations that has to be applied to the data referenced by the URI attribute in order to get the actual representation of the `QualifyingProperties` element. The processing model for the chain of transformations is as defined in clause 4.3.3.2 of XMLDSIG [5].

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

7 Qualifying properties syntax

This clause describes in detail all qualifying properties which have been introduced in clause 5. It provides a rationale for each property as well as its XML Schema definition together with explanatory textual information.

Clause 7.1 summarizes a set of auxiliary structures that will be needed later on, while the remaining clauses corresponds to a certain qualifying property.

Clause 7.2 describes in detail the qualifying properties that can appear in XAdES electronic signatures forms as described in clause 4.

Clause 7.3 describes in detail the qualifying properties that can appear in XAdES-T electronic signatures forms as described in clause 4.

Clause 7.4 describes in detail the qualifying properties referred to validation data that can appear in the XAdES-C form.

Clause 7.5 describes in detail the qualifying properties referred to different time-stamps that can appear in the XAdES-X form.

Clause 7.6 describes in detail the qualifying properties referred to validation data that can appear in the XAdES-X-L form.

Finally, clause 7.7 describes in detail the qualifying properties referred to different time-stamps that can appear in the XAdES-A form.

7.1 Auxiliary syntax

The following three auxiliary XML structures are utilized in several cases in the subsequent clauses.

7.1.1 The AnyType data type

The `AnyType` Schema data type has a content model that allows a sequence of arbitrary XML elements that is of unrestricted length. Additionally, an element of this data type can bear an unrestricted number of arbitrary attributes. It is used throughout the remaining parts of this clause wherever the content of an XML element has been left open.

```
<xsd:complexType name="AnyType" mixed="true">
  <xsd:sequence>
    <xsd:any namespace="##any"/>
  </xsd:sequence>
  <xsd:anyAttribute namespace="##any"/>
</xsd:complexType>
```

7.1.2 The ObjectIdentifierType data type

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

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

```
<xsd:complexType name="ObjectIdentifierType">
  <xsd:sequence>
    <xsd:element name="Identifier" type="IdentifierType"/>
    <xsd:element name="Description" type="xsd:string" minOccurs="0"/>
    <xsd:element name="DocumentationReferences"
     type="DocumentationReferencesType" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

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

- In an XML environment objects are typically identified by means of an Uniform Resource Identifier, URI. In this case, the content of `Identifier` consists of the identifying URI, and the optional `Qualifier` attribute is not specified.
- In ASN.1 an Object IDentifier (OID) is used to identify an object. To support an OID, the content of Identifier consists of an OID, either encoded as Uniform Resource Name (URN) or as Uniform Resource Identifier (URI). The optional `Qualifier` attribute can be used to provide a hint about the applied encoding (values `OIDAsURN` or `OIDAsURI`).

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

```
<xsd:complexType name="IdentifierType">
  <xsd:complexContent>
    <xsd:extension base="xsd:anyURI">
      <xsd:attribute name="Qualifier" type="QualifierType" use="optional"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="QualifierType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="OIDAsURI"/>
    <xsd:enumeration value="OIDAsURN"/>
  </xsd:restriction>
</xsd:simpleType>
```

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

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

```
<xsd:complexType name="DocumentationReferencesType">
  <xsd:sequence maxOccurs="unbounded">
    <xsd:element name="DocumentationReference" type="xsd:anyURI"/>
  </xsd:sequence>
</xsd:complexType>
```

7.1.3 The `EncapsulatedPKIDataType` data type

The `EncapsulatedPKIDataType` is used to incorporate a piece of PKI data into an XML structure whereas the PKI data is encoded using an ASN.1 encoding mechanism. Examples of such PKI data that are widely used at the time being include X509 certificates and revocation lists, OCSP responses, attribute certificates and time-stamps.

```
<xsd:complexType name="EncapsulatedPKIDataType">
  <xsd:complexContent>
    <xsd:extension base="xsd:base64Binary">
      <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The content of this data type is the piece of PKI data, base64 encoded as defined in [5].

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

7.1.4 The `TimeStampType` data type

Time-Stamps shall be used with XML Advanced Electronic Signatures in a number of use cases:

- A XML Advanced Electronic Signature with Time-Stamp (XAdES-T) includes a time-stamp over the XML Advanced Electronic Signature (XAdES) to protect against repudiation in case of a key compromise.
- Two mechanisms are provided for protection against fraudulence in case of a CA key compromise, obtaining the XAdES-X form:
  - A time-stamp only over all certificate and revocation information references of an XML Advanced Electronic Signature with Complete Validation Data (XAdES-C).
  - A time-stamp computed over the signature value, the signature time-stamp and the certificate and revocation information references present in the XML Advanced Electronic Signature with Complete Validation Data (XAdES-C).

- To provide for long term validity of an XML signature, a time-stamp can be applied over an XML Advanced Electronic Signature with Extended Validation Data (XAdES-X-L) to obtain a XAdES-A form. In this case the time-stamp is called an Archive Time-Stamp. Additional time-stamps can be added to this XAdES-A as time goes on.

- Additionally, time-stamps proving that some or all the data objects to be signed have been created before some time can also be added as signed properties to the XAdES.

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

Time-Stamps specified in the present document will be generated on selected parts of the XAdES signature element.

Below follows the schema definition for the data type used for all the time-stamps mentioned above.

```
<xsd:complexType name="TimeStampType">
  <xsd:sequence>
    <xsd:element name="HashDataInfo" type="HashDataInfoType"
     maxOccurs="unbounded"/>
    <xsd:choice>
      <xsd:element name="EncapsulatedTimeStamp"
       type="EncapsulatedPKIDataType"/>
      <xsd:element name="XMLTimeStamp" type="AnyType"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="HashDataInfoType">
  <xsd:sequence>
    <xsd:element name="Transforms" type="ds:TransformsType" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="uri" type="xsd:anyURI" use="required"/>
</xsd:complexType>
```

Each `HashDataInfo` element contains an `uri` attribute referencing a data object and one `ds:Transforms` element indicating the transformations to make to this data object as described in [5].

The sequence of `HashDataInfo` elements will be used to produce the input of the hash computation process whose result will be included in the time-stamp request to be sent to the TSA.

The actual input to the hash computation is obtained as follows. Each data object referenced in the sequence of elements `HashDataInfo` is transformed according the indications of the corresponding `Transforms` element. Once all the referenced data objects have been transformed, the resulting octets are concatenated in the order in which the data objects are referenced.

The time-stamp generated by the TSA can be either an ASN.1 data object (as defined in [16], use `EncapsulatedTimeStamp`), or it can be encoded as XML (use `XMLTimeStamp`). Since at the time being there is no standard for an XML time-stamp, we provide a placeholder for future use.

7.2 Syntax for XAdES form

This clause describes in detail the qualifying properties that can appear in XAdES and XAdES-T advanced electronic signatures forms as described in clause 4.

7.2.1 The `SigningTime` element

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

The XML Schema recommendation [7] defines an XML type `xsd:dateTime` that allows for the inclusion of the required information. This is the type selected for the `SigningTime` element.

This is a signed property that qualifies the whole signature.

An XML electronic signature aligned with the present document MUST contain exactly one `SigningTime` element .

Below follows the Schema definition for this element:

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

7.2.2 The `SigningCertificate` element

According to what has been stated in the Introduction clause, an electronic signature produced in accordance with the present document incorporates: "a commitment that has been explicitly endorsed under a signature policy, at a given time, by a signer **under an identifier**, e.g. a name or a pseudonym, and optionally a role".

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

Many current schemes simply add the certificate after the signed data and thus are subject to various substitution attacks. An example of a substitution attack is a "bad" CA that would issue a certificate to someone with the public key of someone else. If the certificate from the signer was simply appended to the signature and thus not protected by the signature, any one could substitute one certificate by another and the message would appear to be signed by some one else.

In order to counter this kind of attack, the identifier of the certificate has to be protected by the digital signature from the signer.

The `SigningCertificate` property is designed to prevent the simple substitution of the certificate. This property contains references to certificates and digest values computed on them.

The certificate used to verify the signature shall be identified in the sequence; the signature policy may mandate other certificates be present, that may include all the certificates up to the point of trust.

This is a signed property that qualifies the signature.

An XML electronic signature aligned with the present document MUST contain exactly one `SigningCertificate` element .

Below follows the Schema definition:

```
<xsd:element name="SigningCertificate" type="CertIDListType"/>

<xsd:complexType name="CertIDListType">
  <xsd:sequence>
    <xsd:element name="Cert" type="CertIDType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CertIDType">
  <xsd:sequence>
    <xsd:element name="CertDigest" type="DigestAlgAndValueType"/>
    <xsd:element name="IssuerSerial" type="ds:X509IssuerSerialType"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="DigestAlgAndValueType">
  <xsd:sequence>
    <xsd:element name="DigestMethod" type="ds:DigestMethodType"/>
    <xsd:element name="DigestValue" type="ds:DigestValueType"/>
  </xsd:sequence>
</xsd:complexType>
```

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

The element `IssuerSerial` contains the identifier of one of the certificates referenced in the sequence. Should the `ds:X509IssuerSerial` element appear in the signature to denote the same certificate, its value MUST be consistent with the corresponding `IssuerSerial` element.

If the signer uses an attribute certificate to associate a role with the electronic signature, such a certificate MUST be present in the `SignerRole` property.

The element `CertDigest` contains the digest of one of the certificates referenced in the sequence. It contains two elements: `DigestMethod` indicates the digest algorithm and `DigestValue` contains the value of the digest.

7.2.3 The `SignaturePolicyIdentifier` element

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

The signature policy needs to be available in human readable form so that it can be assessed to meet the requirements of the legal and contractual context in which it is being applied.

To facilitate the automatic processing of an electronic signature the parts of the signature policy which specify the electronic rules for the creation and validation of the electronic signature also need to be in a computer processable form.

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

As it has been stated before, any electronic signature claiming alignment with the present document must contain an unambiguous way allowing the identification of the Signature. The present document specifies two alternatives:

- The electronic signature can contain an explicit and unambiguous identifier of a Signature Policy together with a hash value of the signature policy, so it can be verified that the policy selected by the signer is the one being used by the verifier. An explicit signature policy has a globally unique reference, which, in this way, is bound to an electronic signature by the signer as part of the signature calculation. In these cases, for a given explicit signature policy there shall be one definitive form that has a unique binary encoded value. Finally, a signature policy identified in this way may be qualified by additional information.

- Alternatively, the electronic signature can avoid the inclusion of the aforementioned identifier and hash value. This will be possible when the signature policy can be unambiguously derived from the semantics of the type of data object(s) being signed, and some other information, e.g. national laws or private contractual agreements, that mention that a given signature policy must be used for this type of data content. In such cases, the signature will contain a specific empty element indicating that this implied way to identify the signature policy is used instead the identifier and hash value.

The signature policy identifier is a signed property qualifying the signature.

An XML electronic signature aligned with the present document MUST contain exactly one `SignaturePolicyIdentifier` element.

Below follows the Schema definition for this type:

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

<xsd:complexType name="SignaturePolicyIdentifierType">
  <xsd:choice>
    <xsd:element name="SignaturePolicyId" type="SignaturePolicyIdType"/>
    <xsd:element name="SignaturePolicyImplied"/>
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="SignaturePolicyIdType">
  <xsd:sequence>
    <xsd:element name="SigPolicyId" type="ObjectIdentifierType"/>
    <xsd:element ref="ds:Transforms" minOccurs="0"/>
    <xsd:element name="SigPolicyHash" type="DigestAlgAndValueType"/>
    <xsd:element name="SigPolicyQualifiers"
      type="SigPolicyQualifiersListType" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SigPolicyQualifiersListType">
  <xsd:sequence>
    <xsd:element name="SigPolicyQualifier" type="AnyType"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

The `SignaturePolicyId` element will appear when the signature policy is identified using the first alternative. The `SigPolicyId` element contains an identifier that uniquely identifies a specific version of the signature policy. The `SigPolicyHash` element contains the identifier of the hash algorithm and the hash value of the signature policy. The `SigPolicyQualifier` element can contain additional information qualifying the signature policy identifier. The optional `ds:Transforms` element can contain the transformations performed on the signature policy document before computing its hash. The processing model for these transformations is described in [5].

Alternatively, the `SignaturePolicyImplied` element will appear when the second alternative is used. This empty element indicates that the data object(s) being signed and other external data imply the signature policy.

7.2.3.1 Signature Policy qualifiers

Two qualifiers for the signature policy have been identified so far:

- a URL where a copy of the Signature Policy (SP) may be obtained;
- a user notice that should be displayed when the signature is verified.

Below follows the Schema definition for these two elements:

```
<xsd:element name="SPURI" type="xsd:anyURI"/>
<xsd:element name="SPUserNotice" type="SPUserNoticeType"/>

<xsd:complexType name="SPUserNoticeType">
  <xsd:sequence>
    <xsd:element name="NoticeRef" type="NoticeReferenceType" minOccurs="0"/>
    <xsd:element name="ExplicitText" type="xsd:string" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="NoticeReferenceType">
  <xsd:sequence>
    <xsd:element name="Organization" type="xsd:string"/>
    <xsd:element name="NoticeNumbers" type="IntegerListType"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="IntegerListType">
  <xsd:sequence>
    <xsd:element name="int" type="xsd:integer" minOccurs="0"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

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

7.2.4 The `CounterSignature` element

Some electronic signatures may only be valid if they bear more than one signature. This is the case generally when a contract is signed between two parties. The ordering of the signatures may or may not be important, i.e. one may or may not need to be applied before the other.

Several forms of multiple and counter signatures need to be supported, which fall into two basic categories:

- independent signatures;
- embedded signatures.

Independent signatures are parallel signatures where the ordering of the signatures is not important. Therefore an independent signature will not appear as a CounterSignature property of another independent one.

Embedded signatures are applied one after the other and are used where the order the signatures are applied is important. Multiple embedded signatures are supported using the `CounterSignature` unsigned property. Each countersignature is carried in one `Countersignature` element added to the `Signature` element to which the countersignature is applied.

In a qualified `Signature` the contents of the `CounterSignature` element are one or more signatures (i.e. `ds:Signature` elements) of the `SignatureValue` in the qualified `Signature`.

A countersignature can itself be qualified by a `CounterSignature` property. Thus it is possible to construct arbitrarily long series of countersignatures.

This is a unsigned property that qualifies the signature.

Below follows the Schema definition for this element.

```
<xsd:element name="CounterSignature" type="CounterSignatureType" />

<xsd:complexType name="CounterSignatureType">
  <xsd:sequence>
    <xsd:element ref="ds:Signature"/>
  </xsd:sequence>
</xsd:complexType>
```

The next figure shows a countersigned `Signature`.

Figure 5: Use of CounterSignature element

7.2.5 The `DataObjectFormat` element

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

The `DataObjectFormat` element provides information that describes the format of the signed data object. This element MUST be present when it is mandatory to present the signed data object to human users on verification. This is a signed property that qualifies one specific signed data object. In consequence, an XML electronic signature aligned with the present document MAY contain more than one `DataObjectFormat` elements, each one qualifying one signed data object.

Below follows the schema definition for this element.

```
<xsd:element name="DataObjectFormat" type="DataObjectFormatType"/>

<xsd:complexType name="DataObjectFormatType">
  <xsd:sequence>
    <xsd:element name="Description" type="xsd:string" minOccurs="0"/>
    <xsd:element name="ObjectIdentifier" type="ObjectIdentifierType"
     minOccurs="0"/>
    <xsd:element name="MimeType" type="xsd:string" minOccurs="0"/>
    <xsd:element name="Encoding" type="xsd:anyURI" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="ObjectReference" type="xsd:anyURI"
   use="required"/>
</xsd:complexType>
```

The mandatory `ObjectReference` attribute refers to the `Reference` element of the `ds:Signature` corresponding with the data object qualified by this property.

This element can convey:

- Textual information related to the signed data object(s) in element `Description`;
- An identifier indicating the type of the signed data object(s) in element `ObjectIdentifier`;
- An indication of the MIME type of the signed data object(s), in element `MimeType`;
- An indication of the encoding format of the signed data object(s), in element `Encoding`.

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

7.2.6 The `CommitmentTypeIndication` element

According to what has been stated in the Introduction clause, an electronic signature produced in accordance with the present document incorporates: "a **commitment** that has been explicitly endorsed under a signature policy, at a given time, by a signer under an identifier, e.g. a name or a pseudonym, and optionally a role".

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

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

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

If a signature includes a commitment type indication other than one of those recognized under the signature policy the signature shall be treated as invalid.

How commitment is indicated using the semantics of the data object being signed is outside the scope of the present document.

The commitment type may be:

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

The definition of a commitment type includes:

- the object identifier for the commitment;
- a sequence of qualifiers.

The qualifiers can provide more information about the commitment, it could provide, for example, information about the context be it contractual/legal/application specific.

If an electronic signature does not contain a recognized commitment type then the semantics of the electronic signature is dependent on the data object being signed and the context in which it is being used.

This is a signed property that qualifies signed data object(s). In consequence, an XML electronic signature aligned with the present document MAY contain more than one `CommitmentTypeIndication` elements.

Below follows the schema definition for this element.

```
<xsd:element name="CommitmentTypeIndication"
type="CommitmentTypeIndicationType"/>

<xsd:complexType name="CommitmentTypeIndicationType">
  <xsd:sequence>
    <xsd:element name="CommitmentTypeId" type="ObjectIdentifierType"/>
    <xsd:choice>
      <xsd:element name="ObjectReference" type="xsd:anyURI"
        minOccurs="0" maxOccurs="unbounded"/>
      < xsd:element name="AllSignedDataObjects"/>
    </xsd:choice>
    <xsd:element name="CommitmentTypeQualifiers"
     type="CommitmentTypeQualifiersListType" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CommitmentTypeQualifiersListType">
  <xsd:sequence>
    <xsd:element name="CommitmentTypeQualifier"
     type="AnyType" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

The `CommitmentTypeId` element univocally identifies the type of commitment made by the signer. A number of commitments have been already identified in TS 101 733 [1] (and consequently assigned a corresponding OIDs), namely:

- **Proof of origin** indicates that the signer recognizes to have created, approved and sent the signed data object.
- **Proof of receipt** indicates that signer recognizes to have received the content of the signed data object.
- **Proof of delivery** indicates that the TSP providing that indication has delivered a signed data object in a local store accessible to the recipient of the signed data object.
- **Proof of sender** indicates that the entity providing that indication has sent the signed data object (but not necessarily created it).
- **Proof of approval** indicates that the signer has approved the content of the signed data object.
- **Proof of creation** indicates that the signer has created the signed data object (but not necessarily approved, nor sent it).

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

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

7.2.7 The SignatureProductionPlace element

In some transactions the purported place where the signer was at the time of signature creation may need to be indicated. In order to provide this information a new property may be included in the signature.

This property specifies an address associated with the signer at a particular geographical (e.g. city) location.

This is a signed property that qualifies the signer.

An XML electronic signature aligned with the present document MAY contain at most one SignatureProductionPlace element.

Below follows the schema definition for this element.

```
<xsd:element name="SignatureProductionPlace"
type="SignatureProductionPlaceType"/>

<xsd:complexType name="SignatureProductionPlaceType">
  <xsd:sequence>
    <xsd:element name="City" type="xsd:string" minOccurs="0"/>
    <xsd:element name="StateOrProvince" type="xsd:string" minOccurs="0"/>
    <xsd:element name="PostalCode" type="xsd:string" minOccurs="0"/>
    <xsd:element name="CountryName" type="xsd:string" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

7.2.8 The SignerRole element

According to what has been stated in the Introduction clause, an electronic signature produced in accordance with the present document incorporates: "a commitment that has been explicitly endorsed under a signature policy, at a given time, by a signer under an identifier, e.g. a name or a pseudonym, and optionally a **role**".

While the name of the signer is important, the position of the signer within a company or an organization can be even more important. Some contracts may only be valid if signed by a user in a particular role, e.g. a Sales Director. In many cases who the sales Director really is, is not that important but being sure that the signer is empowered by his company to be the Sales Director is fundamental.

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

- using a claimed role name;
- using an attribute certificate containing a *certified* role.

The signer may state his own role without any certificate to corroborate this claim. In which case the claimed role can be added to the signature as a signed qualifying property.

Unlike public key certificates that bind an identifier to a public key, Attribute Certificates bind the identifier of a certificate to some attributes of its owner, like a role. The Attribute Authority will be most of the time under the control of an organization or a company that is best placed to know which attributes are relevant for which individual. The Attribute Authority may use or point to public key certificates issued by any CA, provided that the appropriate trust may be placed in that CA. Attribute Certificates may have various periods of validity. That period may be quite short, e.g. one day. While this requires that a new Attribute Certificate is obtained every day, valid for that day, this can be advantageous since revocation of such certificates may not be needed. When signing, the signer will have to specify which Attribute Certificate it selects.

This is a signed property that qualifies the signer.

An XML electronic signature aligned with the present document MAY contain at most one SignerRole element.

Below follows the Schema definition for this element.

```
<xsd:element name="SignerRole" type="SignerRoleType"/>

<xsd:complexType name="SignerRoleType">
  <xsd:sequence>
    <xsd:element name="ClaimedRoles" type="ClaimedRolesListType"
      minOccurs="0"/>
    <xsd:element name="CertifiedRoles" type="CertifiedRolesListType"
      minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ClaimedRolesListType">
  <xsd:sequence>
    <xsd:element name="ClaimedRole" type="AnyType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CertifiedRolesListType">
  <xsd:sequence>
    <xsd:element name="CertifiedRole" type="EncapsulatedPKIDataType"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

This property contains a sequence of roles that the signer can play (element `SignerRole`). At least one of the two elements `ClaimedRoles` or `CertifiedRoles` must 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 roles use XML.

The `CertifiedRoles` element contains one or more wrapped attribute certificates for the signer.

7.2.9 The `AllDataObjectsTimeStamp` element

The `AllDataObjectsTimeStamp` element contains the time-stamp computed before the signature production, over the sequence formed by ALL the `ds:Reference` elements within the `ds:SignedInfo` referencing whatever the signer wants to sign except the `SignedProperties` element.

The application must compose the `HashDataInfo` element(s) in a way that the data referenced by all `HashDataInfo` together results in the sequence of `ds:Reference` elements described above.

The `AllDataObjectsTimeStamp` element is a signed property.

Several instances of this property from different TSAs can occur within the same XAdES.

Below follows the schema definition for this element.

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

7.2.10 The `IndividualDataObjectsTimeStamp` element

The `IndividualDataObjectsTimeStamp` element contains the time-stamp computed before the signature production, over a sequence formed by SOME `ds:Reference` elements within the `ds:SignedInfo`. Note that this sequence cannot contain a `ds:Reference` computed on the `SignedProperties` element.

The application must compose the `HashDataInfo` element(s) in a way that the data referenced by all `HashDataInfo` together results in the sequence of `ds:Reference` elements mentioned above.

The `IndividualDataObjectsTimeStamp` element is a signed property that qualifies the signed data object(s).

Several instances of this property can occur within the same XAdES.

Below follows the schema definition for this element.

```
<xsd:element name="IndividualDataObjectsTimeStamp" type="TimeStampType"/>
```

7.3 Syntax for XAdES-T form

This clause describes in detail the time-stamps that can appear in the XAdES-T form.

7.3.1 The `SignatureTimeStamp` element

An important property for long standing signatures is that a signature, having been found once to be valid, shall continue to be so months or years later.

A signer, verifier or both may be required to provide on request, proof that a digital signature was created or verified during the validity period of the all the certificates that make up the certificate path. In this case, the signer, verifier or both will also be required to provide proof that all the user and CA certificates used were not revoked when the signature was created or verified.

It would be quite unacceptable, to consider a signature as invalid even if the keys or certificates were later compromised. Thus there is a need to be able to demonstrate that the signature key was valid around the time that the signature was created to provide long term evidence of the validity of a signature.

Time-stamping by a Time-Stamping Authority (TSA) can provide such evidence. A time-stamp is obtained by sending the hash value of the given data to the TSA. The returned time-stamp is a signed data object that contains the hash value, the identity of the TSA, and the time of stamping. This proves that the given data existed *before* the time of stamping.

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

If a recipient wants to hold a valid *electronic* signature he will have to ensure that he has obtained a valid time-stamp for it, before that key (and any key involved in the validation) is revoked. The sooner the time-stamp is obtained after the signing time, the better.

It is important to note that signatures may be generated "off-line" and time-stamped at a later time by anyone, for example by the signer or any recipient interested in the value of the signature. The time-stamp can thus be provided by the signer together with the signed data object, or obtained by the recipient following receipt of the signed data object.

The Signature Validation Policy can specify a maximum acceptable time difference which is allowed between the time indicated in the `SigningTime` element and the time indicated by the `SignatureTimeStamp` element. If this delay is exceeded then the electronic signature shall be considered as invalid.

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

The `SignatureTimeStamp` element is an unsigned property qualifying the signature. A XAdES-T form MAY contain several `SignatureTimeSamp` elements, obtained from different TSAs.

Below follows the schema definition for this element:

```
<xsd:element name="SignatureTimeStamp" type="TimeStampType"/>
```

The `SignatureTimestamp` element contains a single `HashDataInfo` element that refers to the `ds:SignatureValue` element of the XMLDSIG signature. That is, the input for the timestamp hash computation is the `ds:SignatureValue` XML element.

7.4 Syntax for XAdES-C form

This clause describes in detail the additional qualifying properties referred to validation data that can appear in the XAdES-C form.

When dealing with long term electronic signatures, all the data used in the verification (namely, certificate path and revocation information) of such signatures must be stored and conveniently time-stamped as has been stated in clause 4 for arbitration purposes.

In some environments, it can be convenient to add these data to the electronic signature (as unsigned properties) building up a XAdES-A form.

Alternatively, other systems can consider convenient, for the purpose of arbitration, to archive them elsewhere. In such cases each electronic signature must incorporate references to all these data in the XAdES-C form. This format builds up taking XAdES-T signature and incorporating additional data required for validation:

- the sequence of references to the full set of CA certificates that have been used to validate the electronic signature up to (but not including) the signer's certificate;
- a full set of references to the revocation data that have been used in the validation of the signer and CA certificates.

7.4.1 The `CompleteCertificateRefs` element

This clause defines the XML element able to carry the aforementioned references to the certificates: the `CompleteCertificateRefs` element.

This is an unsigned property that qualifies the signature.

An XML electronic signature aligned with the present document MAY contain at most one `CompleteCertificateRefs` element.

Below follows the schema definition for this element.

```
<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>
```

The `CertRefs` element contains a sequence of `Cert` elements already defined in clause 7.2.2, incorporating the digest of each certificate and optionally the issuer and serial number identifier.

7.4.2 The `CompleteRevocationRefs` element

As it was stated in the previous clause, the XAdES-C signatures add to the XAdES-T the full set of references to the revocation data that have been used in the validation of the signer and CAs certificates. They provide means to retrieve the actual revocation data archived elsewhere in case of dispute and, in this way, to illustrate that the verifier has taken due diligence of the available revocation information.

Currently two major types of revocation data are managed in most of the systems, namely CRLs and responses of on-line certificate status servers, obtained through protocols designed for these purposes, like OCSP protocol.

This clause defines the `CompleteRevocationRefs` element that will carry the full set of revocation information used for the verification of the electronic signature.

This is an unsigned property that qualifies the signature.

An XML electronic signature aligned with the present document MAY contain at most one `CompleteRevocationRefs` element.

Below follows the Schema definition for this element.

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

<xsd:complexType name="CompleteRevocationRefsType">
  <xsd:sequence>
    <xsd:element name="CRLRefs" type="CRLRefsType" minOccurs="0"/>
    <xsd:element name="OCSPRefs" type="OCSPRefsType" minOccurs="0"/>
    <xsd:element name="OtherRefs" type="OtherCertStatusRefsType"
minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>

<xsd:complexType name="CRLRefsType">
  <xsd:sequence>
    <xsd:element name="CRLRef" type="CRLRefType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CRLRefType">
  <xsd:sequence>
    <xsd:element name="DigestAlgAndValue" type="DigestAlgAndValueType"/>
    <xsd:element name="CRLIdentifier" type="CRLIdentifierType"
minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CRLIdentifierType">
  <xsd:sequence>
    <xsd:element name="Issuer" type="xsd:string"/>
    <xsd:element name="IssueTime" type="xsd:dateTime" />
    <xsd:element name="Number" type="xsd:integer"  minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
</xsd:complexType>

<xsd:complexType name="OCSPRefsType">
  <xsd:sequence>
    <xsd:element name="OCSPRef" type="OCSPRefType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="OCSPRefType">
  <xsd:sequence>
    <xsd:element name="OCSPIdentifier" type="OCSPIdentifierType"/>
    <xsd:element name="DigestAlgAndValue" type="DigestAlgAndValueType"
      minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="OCSPIdentifierType">
  <xsd:sequence>
    <xsd:element name="ResponderID" type="xsd:string"/>
    <xsd:element name="ProducedAt" type="xsd:dateTime"/>
  </xsd:sequence>
  <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
</xsd:complexType>

<xsd:complexType name="OtherCertStatusRefsType">
  <xsd:sequence>
    <xsd:element name="OtherRef" type="AnyType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

The `CompleteRevocationRefs` element can contain:

- sequences of references to CRLs (`CRLRefs` element);
- sequences of references to OCSP responses (`OCSPRefs` element);
- other references to alternative forms of revocation data (`OtherRefs` element).

Each element in a `CRLRefs` sequence (`CrlRef` element) identifies one CRL. This identification is made by means of:

- the digest of the entire DER encoded (`DigestAlgAndValue` element);
- a set of data (`CRLIdentifier` element) including the issuer (`Issuer` element), the time when the CRL was issued (`IssueTime` element) and optionally the number of the CRL (`Number` element). The `Identifier` element can be dropped if the CRL could be inferred from other information. Its `URI` attribute could serve to indicate where the identified CRL is archived.

Each element in a `OCSPRefs` sequence (`OcspRef` element) identifies one OCSP response. This identification is made by means of:

- a set of data (`OCSPIdentifier` element) including the name of the server that has produced the referenced response (`ResponderID` element) and the time indication in the "ProducedAt" field of the referenced response (`ProducedAt` element). The optional `URI` attribute could serve to indicate where the OCSP response identified is archived;
- the digest computed on the DER encoded OCSP (`DigestAlgAndValue` element) response, since it may be needed to differentiate between two OCSP responses by the same server with their "ProducedAt" fields within the same second.

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

7.5 Syntax for XAdES-X form

This clause describes in detail time-stamps that can appear in the XAdES-X form.

Time-stamped extended electronic signatures are needed when there is a requirement to safeguard against the possibility of a CA key in the certificate chain ever being compromised. A verifier may be required to provide on request, proof that the certification path and the revocation information used a the time of the signature were valid, even in the case where one of the issuing keys or OCSP responder keys is later compromised.

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

- Time-stamp the sequence formed by the digital signature (`ds:Signature` element), the time-stamp(s) present in the XAdES-T form, the certification path references and the revocation status references, when an OCSP response is used to get the status of the certificate from the signer.
- Time-stamp only the certification path and revocation information references when a CRL is used to get the status of the certificate from the signer.

The signer, verifier or both may obtain the time-stamp.

7.5.1 The SigAndRefsTimeStamp element

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 the digital signature (`ds:Signature` element), the time-stamp(s) present in the XAdES-T form, the certification path references and the revocation status references. For the same cryptographic price, this provides an integrity mechanism over the XAdES-C. Any modification can be immediately detected. It should be noticed that other means of protecting/detecting the integrity of the XAdES-C exist and could be used.

The form obtained by the concatenation of successive time-stamps of this type is an XAdES-X (XML Advanced Electronic Signature with extended validation data).

The `SigAndRefsTimeStamp` element is an unsigned property qualifying the signature. A XAdES-X form MAY contain several `SigAndRefsTimeStamp` elements, obtained from different TSAs.

Below follows the schema definition for this element.

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

The `SigAndRefsTimeStamp` element contains the folliowing `HashDataInfo` elements:

- One referring to the `ds:SignatureValue` element of the qualified XMLDSIG signature.
- One per each `SignatureTimeStamp` property element present in XAdES-T.
- One referring to the `CompleteCertificateRefs` property element.
- One referring to the `CompleteRevocationRefs` property element.

That is, the input for the timestamp hash computation is a sequence of the following XML elements:

(`ds:SignatureValue, SignatureTimeStamp+, CompleteCertificateRefs, CompleteRevocationRefs`).

7.5.2 The RefsOnlyTimeStamp element

Time-Stamping each ES with Complete Validation Data 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 of commonly used certificates and CRLs 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 were all the signers use the same CA and the CRL applies for the whole day). The information that needs to be time-stamped is not the actual certificates and CRLs but the unambiguous references to those certificates and CRLs.

The form obtained by the concatenation of successive time-stamps of this type is an XAdES-X (XML Advanced Electronic Signature with extended validation data).

The hash sent to the TSA will be computed then over the concatenation of `CompleteCertificateRefs` and `CompleteRevocationRefs` elements.

The `RefsOnlyTimeStamp` element is an unsigned property qualifying the signature. A XAdES-X form MAY contain several `RefsOnlyTimeStamp` elements, obtained from different TSAs.

Below follows the schema definition for this element.

```
<xsd:element name="RefsOnlyTimeStamp" type="TimeStampType"/>
```

The `SigAndRefsTimeStamp` element contains two `HashDataInfo` elements:

- The first one refers to the `CompleteCertificateRefs` property element.
- The second one refers to the `CompleteRevocationRefs` property element.

That is, the input for the timestamp hash computation is a sequence of the following XML elements:

(`CompleteCertificateRefs, CompleteRevocationRefs`)

7.6 Syntax for XAdES-X-L form

This clause describes in detail the additional qualifying properties referred to validation data that can appear in the XAdES-X-L form.

7.6.1 The `CertificateValues` Property element

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 the "Signature Validation Policy". "Signature Validation Policy" is the term used for the set of rules specifically devoted to the validation process in the Signature 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 of the "Signature validation policy".

When dealing with long term electronic signatures, all the data used in the verification (including the certificate path) must be conveniently archived. The archiving of all this material with the electronic signature gives place to the XAdES-X-L form.

In principle, the `CertificateValues` element contains the full set of certificates that have been used to validate the electronic signature, including the signer's certificate. However, it is not necessary to include one of those certificates into this property, if the certificate is already present in the `ds:KeyInfo` element of the signature.

In fact, both the signer certificate (referenced in the mandatory `SigningCertificate` property element) and all certificates referenced in the `CompleteCertificateRefs` property element must be present either in the `ds:KeyInfo` element of the signature or in the `CertificateValues` property element.

The `CertificateValues` is an unsigned property and qualifies the XML signature.

An XML electronic signature aligned with the present document MAY contain at most one `CertificateValues` element.

```
<xsd:element name="CertificateValues" type="CertificateValuesType"/>

<xsd:complexType name="CertificateValuesType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="EncapsulatedX509Certificate"
          type="EncapsulatedPKIDataType"/>
        <xsd:element name="OtherCertificate" type="AnyType"/>
    </xsd:choice>
    <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
```

The `EncapsulatedX509Certificate` element is able to contain the base64 encoding of a DER-encoded X.509 certificate. The `OtherCertificate` element is a placeholder for potential future new formats of certificates.

7.6.2 The RevocationValues property element

When dealing with long term electronic signatures, all the revocation data used in the verification of such signatures must be stored and conveniently time-stamped as has been stated in clause 4 for arbitration purposes.

Currently two major types of revocation data are managed in most of the systems, namely CRLs and responses of on-line certificate status servers, obtained through protocols designed for these purposes, like OCSP protocol.

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 verification the appropriate certificate revocation information from the signer's CA. This should be done as soon as possible to minimize the time delay between the generation and verification 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 shall be conveyed to the verifier. Additional CRLs for the CA certificates in the certificate path must also be checked by the verifier. It may be convenient to archive these CRLs within an XAdES-A for ease of subsequent verification or arbitration.

When using OCSP to get revocation information, a verifier will have to make sure that she or he gets at the time of the first verification an OCSP response that contains the status "valid". This should be done as soon as possible after the generation of the signature. The signer, the verifier or any other third party may fetch this OCSP response. Since OCSP responses are transient and thus are not archived by any TSP including CA, it is the responsibility of every verifier to make sure that it is stored in a safe place. The simplest way is to store them associated with the electronic signature in its XAdES-X-L form.

The `RevocationValues` property element is used to hold the values of the revocation information which are to be shipped with the XML signature in case of an XML Advanced Electronic Signature with Extended Validation Data (XAdES-X-Long).

This is a unsigned property that qualifies the signature.

An XML electronic signature aligned with the present document MAY contain at most one `RevocationValues` element.

Below follows the Schema definition for this element.

```
<xsd:element name="RevocationValues" type="RevocationValuesType"/>

<xsd:complexType name="RevocationValuesType">
  <xsd:sequence>
    <xsd:element name="CRLValues" type="CRLValuesType" minOccurs="0"/>
    <xsd:element name="OCSPValues" type="OCSPValuesType" minOccurs="0"/>
    <xsd:element name="OtherValues" type="OtherCertStatusValuesType"
minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:ID"  use="optional"/>
</xsd:complexType>
```

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

```
<xsd:complexType name="CRLValuesType">
  <xsd:sequence>
     <xsd:element name="EncapsulatedCRLValue" type="EncapsulatedPKIDataType"
       maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

Certificate Revocation Lists (`CRLValues`) consist of a sequence of at least one Certificate Revocation List. Each EncapsulatedCRLValue will contain the base64 encoding of a DER-encoded X509 CRL.

```
<xsd:complexType name="OCSPValuesType">
  <xsd:sequence>
      <xsd:element name="EncapsulatedOCSPValue"
        type="EncapsulatedPKIDataType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

OCSP Responses (`OCSPValues`) consist of a sequence of at least one OCSP Response. The `EncapsulatedOCSPValue` element contains the base64 encoding of a DER-encoded OCSP Response.

```
<xsd:complexType name="OtherCertStatusValuesType">
  <xsd:sequence>
    <xsd:element name="OtherValue" type="AnyType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

The `OtherValues` element provides a placeholder for other revocation information that can be used in the future. The `ObjectIdentifier` element is used to specify the type of revocation information that is contained by the subsequent `Value` element.

7.7 Syntax for XAdES-A form

This clause describes in detail time-stamps that can appear in the XAdES-A form.

7.7.1 The `ArchiveTimeStamp` element

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

Over a period of time weaknesses may occur in the cryptographic algorithms used to create an electronic signature (e.g. due to the time available for cryptoanalysis, or improvements in cryptoanalytical techniques). Before such weaknesses become likely, a verifier should take extra measures to maintain the validity of the electronic signature. Several techniques could be used to achieve this goal depending on the nature of the weakened cryptography. In order to simplify matters, a single technique, called Archive validation data (XAdES-A form), covering all the cases is being used in the present document.

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

The potential for Trusted Service Provider (TSP) key compromise should be significantly lower than for user keys, because TSP(s) are expected to use stronger cryptography and better key protection. It can be expected that new algorithms (or old ones with greater key lengths) will be used. In such a case, a sequence of time-stamps will protect against forgery. Each time-stamp needs to be affixed before either the compromise of the signing key or of the cracking of the algorithms used by the TSA. TSAs (Time-Stamping Authorities) should have long keys (e.g. which at the time of drafting the present document was 2048 bits for the signing RSA algorithm) and/or a "good" or different algorithm.

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

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

The hash sent to the TSA (messageImprint) will be computed on the XAdES-X-L form of the electronic signature and the signed data objects, ie on the sequence formed as explained below.

The ArchiveTimeStamp element is an unsigned property qualifying the signature. A XAdES-A form MAY contain several ArchiveTimeStamp elements.

Below follows the schema definition for this element.

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

The XAdESArchiveTimeStamp element contains the following sequence of HashDataInfo elements:

- One HashDataInfo element for each data object signed by the XMLDSIG signature. The result of application of the transforms specified each HashDataInfo must be exactly the same as the octet stream that was originally used for computing the digest value of the corresponding ds:Reference.
- One HashDataInfo element for the ds:SignedInfo element. The result of application of the transforms specified in this HashDataInfo must be exactly the same as the octet stream that was originally used for computing the signature value of the XMLDSIG signature.
- One HashDataInfo element for the SignedSignatureProperties element.
- One HashDataInfo element for the SignedDataObjectProperties element.
- One HashDataInfo element for the ds:SignatureValue element.
- One HashDataInfo element per each SignatureTimeStamp property.
- One HashDataInfo element for the CompleteCertificateRefs property.
- One HashDataInfo element for the CompleteRevocationRefs property.
- One HashDataInfo element for the CertificatesValues property (add this property previously if not already present).
- One HashDataInfo element for the RevocationValues property (add this property previously if not already present).
- One HashDataInfo element per each SigAndRefsTimeStamp property (if present).
- One HashDataInfo element per each property RefsOnlyTimeStamp (if present).
- One HashDataInfo element per each any previous XAdESArchiveTimestamp property (if present).

Annex A: Definitions

Data Object (Content/Document) (source W3C Candidate Recommendation August 2001: "XML-Signature Syntax and Processing")

The actual binary/octet data being operated on (transformed, digested or signed) by an application –frequently an HTTP entity. Note that the proper noun Object designates a specific XML element. Occasionally we refer to a data object as a document or as a resource's content. The term elmenent content is used to describe the data between XML start and end tags. The term XML document is used to describe data objects which conform to the XML specification.

Signature (source W3C Candidate Recommendation August 2001: "XML-Signature Syntax and Processing")

Formally speaking, a value generated from the application of a private key to a message via a cryptographic algorithm such that it has the properties of signer authentication and message authentication (integrity). (However, we sometimes use the term signature generically such that it encompasses AuthenticationCode values as well, but we are careful to make the distinction when the property of AuthenticationSigner is relevant to the exposition.) A signature may be (non-exclusively) described as detached, enveloping or enveloped.

Transform (source W3C Candidate Recommendation August 2001: "XML-Signature Syntax and Processing")

The processing of a data from its source to its derived form. Typical transforms include XML Canonicalization, XPath and XSLT.

Attribute Certificate (source ITU-T Recommendation X.509 [9**]:"DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS.DIRECTORY")**

A set of attributes of a user together with some other information, rendered unforgeable by the digital signature created using the private key of the certification authority which issued it.

Annex B (normative): Schema definitions

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://uri.etsi.org/01903/v1.1.1#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://uri.etsi.org/01903/v1.1.1#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" elementFormDefault="qualified">


<!-- Start auxiliary types definitions: AnyType, ObjectIdentifierType,
EncapsulatedPKIDataType and TimestampType-->

<!-- Start AnyType -->

  <xsd:element name="Any" type="AnyType"/>
  <xsd:complexType name="AnyType" mixed="true">
    <xsd:sequence>
      <xsd:any namespace="##any"/>
    </xsd:sequence>
    <xsd:anyAttribute namespace="##any"/>
  </xsd:complexType>

<!-- End AnyType -->

<!-- Start ObjectIdentifierType-->

  <xsd:element name="ObjectIdentifier" type="ObjectIdentifierType"/>
  <xsd:complexType name="ObjectIdentifierType">
    <xsd:sequence>
      <xsd:element name="Identifier" type="IdentifierType"/>
      <xsd:element name="Description" type="xsd:string" minOccurs="0"/>
      <xsd:element name="DocumentationReferences"
type="DocumentationReferencesType" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="IdentifierType">
    <xsd:complexContent>
      <xsd:extension base="xsd:anyURI">
        <xsd:attribute name="Qualifier" type="QualifierType"
use="optional"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:simpleType name="QualifierType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="OIDAsURI"/>
      <xsd:enumeration value="OIDAsURN"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType name="DocumentationReferencesType">
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="DocumentationReference" type="xsd:anyURI"/>
    </xsd:sequence>
  </xsd:complexType>

<!-- End ObjectIdentifierType-->

<!-- Start EncapsulatedPKIDataType-->

  <xsd:element name="EncapsulatedPKIData" type="EncapsulatedPKIDataType"/>
  <xsd:complexType name="EncapsulatedPKIDataType">
    <xsd:complexContent>
      <xsd:extension base="xsd:base64Binary">
        <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
      </xsd:extension>
    </xsd:complexContent>
```

```
  </xsd:complexType>

<!-- End EncapsulatedPKIDataType -->

<!-- Start TimeStampType -->

  <xsd:element name="TimeStamp" type="TimeStampType"/>
  <xsd:complexType name="TimeStampType">
    <xsd:sequence>
      <xsd:element name="HashDataInfo" type="HashDataInfoType"
maxOccurs="unbounded"/>
      <xsd:choice>
        <xsd:element name="EncapsulatedTimeStamp"
type="EncapsulatedPKIDataType"/>
        <xsd:element name="XMLTimeStamp" type="AnyType"/>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="HashDataInfoType">
    <xsd:sequence>
      <xsd:element name="Transforms" type="ds:TransformsType"
minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="uri" type="xsd:anyURI" use="required"/>
  </xsd:complexType>

<!-- End TimeStampType -->

<!-- End auxiliary types definitions-->

<!-- Start container types -->

<!-- Start QualifyingProperties -->

  <xsd:element name="QualifyingProperties" type="QualifyingPropertiesType"/>

  <xsd:complexType name="QualifyingPropertiesType">
    <xsd:sequence>
      <xsd:element name="SignedProperties" type="SignedPropertiesType"
minOccurs="0"/>
      <xsd:element name="UnsignedProperties" type="UnsignedPropertiesType"
minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="Target" type="xsd:anyURI" use="required"/>
    <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
  </xsd:complexType>

<!-- End QualifyingProperties -->

<!-- Start SignedProperties-->

  <xsd:element name="SignedProperties" type="SignedPropertiesType"/>

  <xsd:complexType name="SignedPropertiesType">
    <xsd:sequence>
      <xsd:element name="SignedSignatureProperties"
type="SignedSignaturePropertiesType"/>
      <xsd:element name="SignedDataObjectProperties"
type="SignedDataObjectPropertiesType" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
  </xsd:complexType>

<!-- End SignedProperties-->

<!-- Start UnsignedProperties-->

<xsd:element name="UnsignedProperties" type="UnsignedPropertiesType" />

  <xsd:complexType name="UnsignedPropertiesType">
    <xsd:sequence>
```

```
      <xsd:element name="UnsignedSignatureProperties"
        type="UnsignedSignaturePropertiesType" minOccurs="0"/>
        <xsd:element name="UnsignedDataObjectProperties"
          type="UnsignedDataObjectPropertiesType" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
  </xsd:complexType>

<!-- End UnsignedProperties-->

<!-- Start SignedSignatureProperties-->

<xsd:element name="SignedSignatureProperties"
  type="SignedSignaturePropertiesType" />

<xsd:complexType name="SignedSignaturePropertiesType">
  <xsd:sequence>
    <xsd:element name="SigningTime" type="xsd:dateTime"/>
    <xsd:element name="SigningCertificate" type="CertIDListType"/>
    <xsd:element name="SignaturePolicyIdentifier"
     type="SignaturePolicyIdentifierType"/>
    <xsd:element name="SignatureProductionPlace"
type="SignatureProductionPlaceType"
     minOccurs="0"/>
    <xsd:element name="SignerRole" type="SignerRoleType" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<!-- End SignedSignatureProperties-->


<!-- Start SignedDataObjectProperties-->

<xsd:element name="SignedDataObjectProperties"
  type="SignedDataObjectPropertiesType"/>

<xsd:complexType name="SignedDataObjectPropertiesType">
  <xsd:sequence>
    <xsd:element name="DataObjectFormat" type="DataObjectFormatType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="CommitmentTypeIndication"
      type="CommitmentTypeIndicationType" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="AllDataObjectsTimeStamp" type="TimeStampType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="IndividualDataObjectsTimeStamp" type="TimeStampType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<!-- End SignedDataObjectProperties-->


<!-- Start UnsignedSignatureProperties-->

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

<xsd:complexType name="UnsignedSignaturePropertiesType">
  <xsd:sequence>
    <xsd:element name="CounterSignature" type="CounterSignatureType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="SignatureTimeStamp" type="TimeStampType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="CompleteCertificateRefs"
      type="CompleteCertificateRefsType" minOccurs="0"/>
    <xsd:element name="CompleteRevocationRefs"
      type="CompleteRevocationRefsType" minOccurs="0"/>
    <xsd:choice>
      <xsd:element name="SigAndRefsTimeStamp" type="TimeStampType"
        minOccurs="0" maxOccurs="unbounded"/>
```

```
      <xsd:element name="RefsOnlyTimeStamp" type="TimeStampType"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:choice>
    <xsd:element name="CertificateValues" type="CertificateValuesType"
      minOccurs="0"/>
    <xsd:element name="RevocationValues" type="RevocationValuesType"
      minOccurs="0"/>
    <xsd:element name="ArchiveTimeStamp" type="TimeStampType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  </xsd:complexType>


<!-- End UnsignedSignatureProperties-->


<!-- Start UnsignedDataObjectProperties-->

<xsd:element name="UnsignedDataObjectProperties"
  type="UnsignedDataObjectPropertiesType" />

<xsd:complexType name="UnsignedDataObjectPropertiesType">
  <xsd:sequence>
    <xsd:element name="UnsignedDataObjectProperty" type="AnyType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<!-- End UnsignedDataObjectProperties-->

<!-- Start QualifyingPropertiesReference-->

<xsd:element name="QualifyingPropertiesReference"
  type="QualifyingPropertiesReferenceType"/>

<xsd:complexType name="QualifyingPropertiesReferenceType">
  <xsd:sequence>
    <xsd:element name="Transforms" type="ds:TransformsType" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="URI" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>

<!-- End QualifyingPropertiesReference-->

<!-- End container types -->


<!-- Start SigningTime element -->
  <xsd:element name="SigningTime" type="xsd:dateTime"/>

<!-- End SigningTime element -->

<!-- Start SigningCertificate -->
  <xsd:element name="SigningCertificate" type="CertIDListType"/>
  <xsd:complexType name="CertIDListType">
    <xsd:sequence>
      <xsd:element name="Cert" type="CertIDType" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CertIDType">
    <xsd:sequence>
      <xsd:element name="CertDigest" type="DigestAlgAndValueType"/>
      <xsd:element name="IssuerSerial" type="ds:X509IssuerSerialType"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="DigestAlgAndValueType">
    <xsd:sequence>
      <xsd:element name="DigestMethod" type="ds:DigestMethodType"/>
      <xsd:element name="DigestValue" type="ds:DigestValueType"/>
    </xsd:sequence>
```

```
  </xsd:complexType>

<!-- End SigningCertificate -->

<!-- Start SignaturePolicyIdentifier -->

  <xsd:element name="SignaturePolicyIdentifier"
type="SignaturePolicyIdentifierType"/>
  <xsd:complexType name="SignaturePolicyIdentifierType">
    <xsd:choice>
      <xsd:element name="SignaturePolicyId" type="SignaturePolicyIdType"/>
      <xsd:element name="SignaturePolicyImplied"/>
    </xsd:choice>
  </xsd:complexType>
  <xsd:complexType name="SignaturePolicyIdType">
    <xsd:sequence>
      <xsd:element name="SigPolicyId" type="ObjectIdentifierType"/>
      <xsd:element ref="ds:Transforms" minOccurs="0"/>
      <xsd:element name="SigPolicyHash" type="DigestAlgAndValueType"/>
      <xsd:element name="SigPolicyQualifiers"
type="SigPolicyQualifiersListType" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="SigPolicyQualifiersListType">
    <xsd:sequence>
      <xsd:element name="SigPolicyQualifier" type="AnyType"
maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="SPURI" type="xsd:anyURI"/>
  <xsd:element name="SPUserNotice" type="SPUserNoticeType"/>
  <xsd:complexType name="SPUserNoticeType">
    <xsd:sequence>
      <xsd:element name="NoticeRef" type="NoticeReferenceType"
minOccurs="0"/>
      <xsd:element name="ExplicitText" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="NoticeReferenceType">
    <xsd:sequence>
      <xsd:element name="Organization" type="xsd:string"/>
      <xsd:element name="NoticeNumbers" type="IntegerListType"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="IntegerListType">
    <xsd:sequence>
      <xsd:element name="int" type="xsd:integer" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

<!-- End SignaturePolicyIdentifier -->


<!-- Start CounterSignature -->
  <xsd:element name="CounterSignature" type="CounterSignatureType"/>
  <xsd:complexType name="CounterSignatureType">
    <xsd:sequence>
      <xsd:element ref="ds:Signature"/>
    </xsd:sequence>
  </xsd:complexType>

<!-- End CounterSignature -->

<!-- Start DataObjectFormat -->

  <xsd:element name="DataObjectFormat" type="DataObjectFormatType"/>
  <xsd:complexType name="DataObjectFormatType">
    <xsd:sequence>
      <xsd:element name="Description" type="xsd:string" minOccurs="0"/>
```

```
       <xsd:element name="ObjectIdentifier" type="ObjectIdentifierType"
minOccurs="0"/>
       <xsd:element name="MimeType" type="xsd:string" minOccurs="0"/>
       <xsd:element name="Encoding" type="xsd:anyURI" minOccurs="0"/>
     </xsd:sequence>
     <xsd:attribute name="ObjectReference" type="xsd:anyURI" use="required"/>
   </xsd:complexType>

<!-- End DataObjectFormat -->

<!-- Start CommitmentTypeIndication -->

   <xsd:element name="CommitmentTypeIndication"
type="CommitmentTypeIndicationType"/>
   <xsd:complexType name="CommitmentTypeIndicationType">
     <xsd:sequence>
       <xsd:element name="CommitmentTypeId" type="ObjectIdentifierType"/>
       <xsd:choice>
         <xsd:element name="ObjectReference" type="xsd:anyURI" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element name="AllSignedDataObjects"/>
       </xsd:choice>
       <xsd:element name="CommitmentTypeQualifiers"
type="CommitmentTypeQualifiersListType" minOccurs="0"/>
     </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="CommitmentTypeQualifiersListType">
     <xsd:sequence>
       <xsd:element name="CommitmentTypeQualifier" type="AnyType"
minOccurs="0" maxOccurs="unbounded"/>
     </xsd:sequence>
   </xsd:complexType>

<!-- End CommitmentTypeIndication -->

<!-- Start SignatureProductionPlace -->

   <xsd:element name="SignatureProductionPlace"
type="SignatureProductionPlaceType"/>
   <xsd:complexType name="SignatureProductionPlaceType">
     <xsd:sequence>
       <xsd:element name="City" type="xsd:string" minOccurs="0"/>
       <xsd:element name="StateOrProvince" type="xsd:string" minOccurs="0"/>
       <xsd:element name="PostalCode" type="xsd:string" minOccurs="0"/>
       <xsd:element name="CountryName" type="xsd:string" minOccurs="0"/>
     </xsd:sequence>
   </xsd:complexType>

<!-- End SignatureProductionPlace -->

<!-- Start SignerRole -->

<xsd:element name="SignerRole" type="SignerRoleType"/>
<xsd:complexType name="SignerRoleType">
  <xsd:sequence>
    <xsd:element name="ClaimedRoles" type="ClaimedRolesListType"
      minOccurs="0"/>
    <xsd:element name="CertifiedRoles" type="CertifiedRolesListType"
      minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ClaimedRolesListType">
  <xsd:sequence>
    <xsd:element name="ClaimedRole" type="AnyType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CertifiedRolesListType">
  <xsd:sequence>
    <xsd:element name="CertifiedRole" type="EncapsulatedPKIDataType"
```

```
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<!-- End SignerRole -->


   <xsd:element name="AllDataObjectsTimeStamp" type="TimeStampType"/>

   <xsd:element name="IndividualDataObjectsTimeStamp" type="TimeStampType"/>

   <xsd:element name="SignatureTimeStamp" type="TimeStampType"/>

<!-- Start CompleteCertificateRefs -->

<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>

<!-- End CompleteCertificateRefs -->


<!-- Start CompleteRevocationRefs-->

<xsd:element name="CompleteRevocationRefs"
type="CompleteRevocationRefsType"/>

<xsd:complexType name="CompleteRevocationRefsType">
  <xsd:sequence>
    <xsd:element name="CRLRefs" type="CRLRefsType" minOccurs="0"/>
    <xsd:element name="OCSPRefs" type="OCSPRefsType" minOccurs="0"/>
    <xsd:element name="OtherRefs" type="OtherCertStatusRefsType"
minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>

<xsd:complexType name="CRLRefsType">
  <xsd:sequence>
    <xsd:element name="CRLRef" type="CRLRefType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CRLRefType">
  <xsd:sequence>
    <xsd:element name="DigestAlgAndValue" type="DigestAlgAndValueType"/>
    <xsd:element name="CRLIdentifier" type="CRLIdentifierType"
minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CRLIdentifierType">
  <xsd:sequence>
    <xsd:element name="Issuer" type="xsd:string"/>
    <xsd:element name="IssueTime" type="xsd:dateTime" />
    <xsd:element name="Number" type="xsd:integer"  minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
</xsd:complexType>

<xsd:complexType name="OCSPRefsType">
  <xsd:sequence>
    <xsd:element name="OCSPRef" type="OCSPRefType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="OCSPRefType">
  <xsd:sequence>
    <xsd:element name="OCSPIdentifier" type="OCSPIdentifierType"/>
    <xsd:element name="DigestAlgAndValue" type="DigestAlgAndValueType"
      minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="OCSPIdentifierType">
  <xsd:sequence>
    <xsd:element name="ResponderID" type="xsd:string"/>
    <xsd:element name="ProducedAt" type="xsd:dateTime"/>
  </xsd:sequence>
  <xsd:attribute name="URI" type="xsd:anyURI" use="optional"/>
</xsd:complexType>

<xsd:complexType name="OtherCertStatusRefsType">
  <xsd:sequence>
    <xsd:element name="OtherRef" type="AnyType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<!-- End CompleteRevocationRefs-->


<xsd:element name="SigAndRefsTimeStamp" type="TimeStampType"/>

<xsd:element name="RefsOnlyTimeStamp" type="TimeStampType"/>

<!-- Start CertificateValues -->

<xsd:element name="CertificateValues" type="CertificateValuesType"/>

<xsd:complexType name="CertificateValuesType">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="EncapsulatedX509Certificate"
type="EncapsulatedPKIDataType"/>
     <xsd:element name="OtherCertificate" type="AnyType"/>
    </xsd:choice>
    <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>

<!-- End CertificateValues -->

<!-- Start RevocationValues-->

<xsd:element name="RevocationValues" type="RevocationValuesType"/>

<xsd:complexType name="RevocationValuesType">
  <xsd:sequence>
    <xsd:element name="CRLValues" type="CRLValuesType" minOccurs="0"/>
    <xsd:element name="OCSPValues" type="OCSPValuesType" minOccurs="0"/>
    <xsd:element name="OtherValues" type="OtherCertStatusValuesType"
minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>

<xsd:complexType name="CRLValuesType">
  <xsd:sequence>
     <xsd:element name="EncapsulatedCRLValue" type="EncapsulatedPKIDataType"
       maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="OCSPValuesType">
  <xsd:sequence>
      <xsd:element name="EncapsulatedOCSPValue"
        type="EncapsulatedPKIDataType" maxOccurs="unbounded"/>
  </xsd:sequence>
```

```
</xsd:complexType>
<xsd:complexType name="OtherCertStatusValuesType">
  <xsd:sequence>
    <xsd:element name="OtherValue" type="AnyType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<!-- End RevocationValues-->

<xsd:element name="ArchiveTimeStamp" type="TimeStampType"/>


</xsd:schema>
```

Annex C:
DTD

```
<?xml version="1.0" encoding="UTF-8"?>


<!ENTITY % Any.ANY ''>
<!ENTITY % XMLTimeStamp.ANY ''>

 <!-- Start Any -->

<!ELEMENT Any (#PCDATA  %Any.ANY;)*>

<!-- End Any -->

<!-- Start ObjectIdentifier -->

<!ELEMENT ObjectIdentifier (Identifier, Description?,
DocumentationReferences?)>
<!ELEMENT Identifier (#PCDATA)>
<!ATTLIST Identifier
  Qualifier (OIDAsURI | OIDAsURN) #IMPLIED
>
<!ELEMENT Description (#PCDATA)>
<!ELEMENT DocumentationReferences (DocumentationReference)+>
<!ELEMENT DocumentationReference (#PCDATA)>

<!-- End ObjectIdentifier -->

<!-- Start EncapsulatedPKIData -->

<!ELEMENT EncapsulatedPKIData (#PCDATA)>
<!ATTLIST EncapsulatedPKIData
  Id ID #IMPLIED
>

<!-- End EncapsulatedPKIData -->

<!-- Start EncapsulatedTimeStamp -->

<!ELEMENT TimeStamp (HashDataInfo+, (EncapsulatedTimeStamp | XMLTimeStamp))>

<!ELEMENT HashDataInfo (Transforms?)>
<!ATTLIST HashDataInfo
  uri CDATA #REQUIRED
>

<!ELEMENT EncapsulatedTimeStamp (#PCDATA)>
<!ATTLIST EncapsulatedTimeStamp
  Id ID #IMPLIED
>
<!ELEMENT XMLTimeStamp (#PCDATA %XMLTimeStamp.ANY;)*>

<!-- End EncapsulatedTimeStamp -->

<!-- Start container types -->

<!-- Start QualifyingProperties -->

<!ELEMENT QualifyingProperties (SignedProperties, UnsignedProperties?)>
<!ATTLIST QualifyingProperties
  Target CDATA #REQUIRED
  Id ID #IMPLIED
>

<!ELEMENT SignedProperties (SignedSignatureProperties,
SignedDataObjectProperties?)>
```

```
<!ATTLIST SignedProperties
  Id ID #IMPLIED
>
<!ELEMENT UnsignedProperties (UnsignedSignatureProperties?,
UnsignedDataObjectProperties?)>
<!ATTLIST UnsignedProperties
  Id ID #IMPLIED
>

<!-- End QualifyingProperties -->

<!-- Start SignedSignatureProperties, SignedDataObjectProperties,
UnsignedSignatureProperties, UnsignedDataObjectProperties -->

<!ELEMENT SignedSignatureProperties (SigningTime, SigningCertificate,
SignaturePolicyIdentifier, SignatureProductionPlace?, SignerRole?)>
<!ELEMENT SignedDataObjectProperties (DataObjectFormat*,
CommitmentTypeIndication*, AllDataObjectsTimeStamp*,
IndividualDataObjectsTimeStamp*)>

<!ELEMENT UnsignedSignatureProperties (CounterSignature*,
SignatureTimeStamp*, CompleteCertificateRefs?, CompleteRevocationRefs?,
(SigAndRefsTimeStamp* | RefsOnlyTimeStamp*), CertificateValues?,
RevocationValues?, ArchiveTimeStamp*)>
<!ELEMENT UnsignedDataObjectProperties (UnsignedDataObjectProperty*)>

<!ELEMENT UnsignedDataObjectProperty (#PCDATA %Any.ANY; )*>

<!-- End SignedSignatureProperties, SignedDataObjectProperties,
UnsignedSignatureProperties, UnsignedDataObjectProperties -->

<!-- Start QualifyingPropertiesReference -->

<!ELEMENT QualifyingPropertiesReference (Transforms?)>
<!ATTLIST QualifyingPropertiesReference
  URI CDATA #REQUIRED
  Id ID #IMPLIED
>

<!-- End QualifyingPropertiesReference -->

<!-- End container types -->

<!-- Start SigningTime -->

<!ELEMENT SigningTime (#PCDATA)>

<!-- End SigningTime -->

<!-- Start SigningCertificate -->

<!ELEMENT SigningCertificate (Cert+)>
<!ELEMENT Cert (CertDigest, IssuerSerial)>
<!ELEMENT CertDigest (DigestMethod, DigestValue)>
<!ELEMENT IssuerSerial (X509IssuerName, X509SerialNumber)>

<!-- End SigningCertificate -->

<!-- Start SignaturePolicyIdentifier -->

<!ELEMENT SignaturePolicyIdentifier (SignaturePolicyId |
SignaturePolicyImplied)>

<!ELEMENT SignaturePolicyId (SigPolicyId, Transforms?, SigPolicyHash,
SigPolicyQualifiers?)>
<!ELEMENT SignaturePolicyImplied ANY>

<!ELEMENT SigPolicyId (Identifier, Description?, DocumentationReferences?)>
<!ELEMENT SigPolicyHash (DigestMethod, DigestValue)>

<!ELEMENT SigPolicyQualifiers (SigPolicyQualifier+)>
```

```
<!ELEMENT SigPolicyQualifier (#PCDATA %Any.ANY; )*>

<!-- End SignaturePolicyIdentifier -->

<!-- Start SPURI and SPUserNotice -->

<!ELEMENT SPURI (#PCDATA)>
<!ELEMENT SPUserNotice (NoticeRef?, ExplicitText?)>

<!ELEMENT NoticeRef (Organization, NoticeNumbers)>
<!ELEMENT ExplicitText (#PCDATA)>

<!ELEMENT Organization (#PCDATA)>
<!ELEMENT NoticeNumbers (#PCDATA)*>

<!-- End SPURI and SPUserNotice -->

<!-- Start CounterSignature -->

<!ELEMENT CounterSignature (Signature)>

<!-- End CounterSignature -->

<!-- Start DataObjectFormat -->

<!ELEMENT DataObjectFormat (Description?, ObjectIdentifier?, MimeType?,
Encoding?)>
<!ATTLIST DataObjectFormat
  ObjectReference CDATA #REQUIRED
>

<!ELEMENT MimeType (#PCDATA)>
<!ELEMENT Encoding (#PCDATA)>

<!-- End DataObjectFormat -->

<!-- Start CommitmentTypeIndication -->

<!ELEMENT CommitmentTypeIndication (CommitmentTypeId, (ObjectReference* |
AllSignedDataObjects), CommitmentTypeQualifiers?)>

<!ELEMENT CommitmentTypeId (Identifier, Description?,
DocumentationReferences?)>
<!ELEMENT ObjectReference (#PCDATA)>
<!ELEMENT AllSignedDataObjects ANY>
<!ELEMENT CommitmentTypeQualifiers (CommitmentTypeQualifier*)>

<!ELEMENT CommitmentTypeQualifier (#PCDATA%Any.ANY; )*>

<!-- End CommitmentTypeIndication -->

<!-- Start SignatureProductionPlace -->

<!ELEMENT SignatureProductionPlace (City?, StateOrProvince?, PostalCode?,
CountryName?)>

<!ELEMENT City (#PCDATA)>
<!ELEMENT StateOrProvince (#PCDATA)>
<!ELEMENT PostalCode (#PCDATA)>
<!ELEMENT CountryName (#PCDATA)>

<!-- End SignatureProductionPlace -->

<!-- Start SignerRole -->

<!-- Start SignerRole -->

<!ELEMENT SignerRole (ClaimedRoles?, CertifiedRoles?)>

<!ELEMENT ClaimedRoles (ClaimedRole+)>
<!ELEMENT CertifiedRoles (CertifiedRole+)>
```

```
<!ELEMENT ClaimedRole (#PCDATA %Any.ANY; )*>
<!ELEMENT CertifiedRole (#PCDATA)>
<!ATTLIST CertifiedRole
  Id ID #IMPLIED
>
<!-- End SignerRole -->

<!-- Start AllDataObjectsTimeStamp, IndividualDataObjectsTimeStamp,
SignatureTimeStamp -->

<!ELEMENT AllDataObjectsTimeStamp (HashDataInfo+, (EncapsulatedTimeStamp |
XMLTimeStamp))>

<!ELEMENT IndividualDataObjectsTimeStamp (HashDataInfo+,
(EncapsulatedTimeStamp | XMLTimeStamp))>

<!ELEMENT SignatureTimeStamp (HashDataInfo+, (EncapsulatedTimeStamp |
XMLTimeStamp))>

<!-- End AllDataObjectsTimeStamp, IndividualDataObjectsTimeStamp,
SignatureTimeStamp -->

<!-- Start CompleteCertificateRefs -->

<!ELEMENT CompleteCertificateRefs (CertRefs)>
<!ATTLIST CompleteCertificateRefs
  Id ID #IMPLIED
>

<!ELEMENT CertRefs (Cert+)>

<!-- End CompleteCertificateRefs -->

<!-- Start CompleteRevocationRefs -->

<!ELEMENT CompleteRevocationRefs (CRLRefs?, OCSPRefs?, OtherRefs?)>
<!ATTLIST CompleteRevocationRefs
  Id ID #IMPLIED
>

<!ELEMENT CRLRefs (CRLRef+)>
<!ELEMENT OCSPRefs (OCSPRef+)>
<!ELEMENT OtherRefs (OtherRef+)>

<!ELEMENT CRLRef (DigestAlgAndValue,CRLIdentifier?)>
<!ELEMENT OCSPRef (OCSPIdentifier, DigestAlgAndValue?)>
<!ELEMENT OtherRef (#PCDATA %Any.ANY; )*>

<!ELEMENT DigestAlgAndValue (DigestMethod, DigestValue)>
<!ELEMENT CRLIdentifier (Issuer, IssueTime, Number?)>
<!ATTLIST Identifier
  URI CDATA #IMPLIED
>
<!ELEMENT OCSPIdentifier (ResponderID, ProducedAt)>
<!ATTLIST Identifier
  URI CDATA #IMPLIED
>

<!ELEMENT Issuer (#PCDATA)>
<!ELEMENT IssueTime (#PCDATA)>
<!ELEMENT Number (#PCDATA)>
<!ELEMENT ResponderID (#PCDATA)>
<!ELEMENT ProducedAt (#PCDATA)>

<!-- End CompleteRevocationRefs -->

<!-- Start SigAndRefsTimeStamp, RefsOnlyTimeStamp  -->

<!ELEMENT SigAndRefsTimeStamp (HashDataInfo+, (EncapsulatedTimeStamp |
XMLTimeStamp))>
```

```
<!ELEMENT RefsOnlyTimeStamp (HashDataInfo+, (EncapsulatedTimeStamp |
XMLTimeStamp))>

<!-- End SigAndRefsTimeStamp, RefsOnlyTimeStamp  -->

<!-- Start CertificateValues -->

<!ELEMENT CertificateValues (EncapsulatedX509Certificate |
OtherCertificate)*>
<!ATTLIST CertificateValues
   Id ID #IMPLIED
>

<!ELEMENT EncapsulatedX509Certificate (#PCDATA)>
<!ATTLIST EncapsulatedX509Certificate
   Id ID #IMPLIED
>
<!ELEMENT OtherCertificate (#PCDATA %Any.ANY; )*>

<!-- End CertificateValues -->

<!-- Start RevocationValues -->

<!ELEMENT RevocationValues (CRLValues?, OCSPValues?, OtherValues?)>
<!ATTLIST RevocationValues
   Id ID #IMPLIED
>

<!ELEMENT CRLValues (EncapsulatedCRLValue+)>
<!ELEMENT OCSPValues (EncapsulatedOCSPValue+)>
<!ELEMENT OtherValues (OtherValue+)>

<!ELEMENT EncapsulatedCRLValue (#PCDATA)>
<!ATTLIST EncapsulatedCRLValue
   Id ID #IMPLIED
>
<!ELEMENT EncapsulatedOCSPValue (#PCDATA)>
<!ATTLIST EncapsulatedOCSPValue
   Id ID #IMPLIED
>
<!ELEMENT OtherValue (#PCDATA%Any.ANY; )*>

<!-- End RevocationValues -->

<!-- Start ArchiveTimeStamp -->

<!ELEMENT ArchiveTimeStamp (HashDataInfo+, (EncapsulatedTimeStamp |
XMLTimeStamp))>

<!-- End ArchiveTimeStamp -->
```

Annex D:
Incorporation of qualifying properties

As stated in the normative part of the present document, new elements have been defined to incorporate properties (both signed and unsigned) that qualify the whole signature, the signer or individual signed data objects: `QualifyingProperties`, `SignedProperties`, `UnsignedProperties`, `SignedSignatureProperties`, `UnsignedSignatureProperties`, `SignedDataObjectProperties` and `UnsignedDataProperties`.

This annex shows an example of direct incorporation of qualifying properties and one example of indirect incorporation of these properties.

Below follows the resulting general structure of direct incorporation.

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

    <SignedProperties>

      <SignedSignatureProperties>
        <!-- Collection of signed XML elements with
        properties qualifying the signature or the
        signer -->
      </SignedSignatureProperties>

      <SignedDataObjectProperties>
        <!-- Collection of signed XML elements with
        properties individually qualifying signed data
        objects -->
         </SignedDataObjectPropertiesSigned>

    </SignedProperties>

    <UnsignedProperties>

      </UnsignedSignatureProperties>
        <!-- Collection of unsigned XML elements with
        properties qualifying signature or signer -->
      </UnsignedSignatureProperties>

      <UnSignedDataObjectProperties>
        <!-- Collection of signed XML elements with
        properties individually qualifying signed
        data objects -->
      </UnSignedDataObjectProperties>

    </UnsignedProperties>

  </ds:Object>
```

```
</ds:Signature>
```

Below follows an example showing the inclusion of three sets of qualifying properties:

- The first one includes signed properties qualifying the signature, namely:
  - the time of signature production (element `SigningTime`);
  - a restricted set of references to certificates to be used in verifying a signature. This also includes a reference to the certificate containing the public key corresponding to the private key used in the signature computation (element `SigningCertificate`);
  - an identification of the signature policy under which the signature has been produced and will have to be verified (element `SignaturePolicyIdentifier`).
- The second one includes signed properties qualifying the signed data object, namely:
  - a time-stamp of the signed data object, proving that the content has been produced before the time indicated in the time-stamp (element `AllDataObjectsTimeStamp`);
  - an indication of the format of the signed object (element `DataObjectFormat`).
- The third one includes unsigned properties qualifying the signature, namely:
  - a time-stamp on the electronic signature itself, proving that the signature was produced before the time indicated by such time-stamp (element `SignatureTimeStamp`);
  - the references to the full set of CA certificates that the verifier of the electronic signature has used to validate the electronic signature (element `CompleteCertificateRefs`);
  - the references to the revocation material (CRLs or OCSP responses) used in the validation of the signer and CA certificates used the full to validate the electronic signature (element `CompleteRevocationRefs`);
  - the time-stamp generated over the electronic signature with the aforementioned qualifying information (element `SigAndRefsTimeStamp`);
  - the full set of CA certificates that the verifier of the electronic signature has used to validate the electronic signature (element `CertificateValues`);
  - the revocation material (CRLs or OCSP responses) used in the validation of the signer and CA certificates used the full to validate the electronic signature (element `RevocationValues`).

```
[s01]<ds:Signature Id="SignatureWithSignedAndUnsignedProperties"
xmlns="http://www.w3.org/2000/09/xmldsig#>
[s02]  <ds:SignedInfo>
[s03]    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2000/WD-
xml-c14n-20000710"/>
[s04]    <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
[s05]    <ds:Reference URI="http://www.etsi.org/docToBeSigned"
Id="FirstSignedDocument">
[s06]      <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha"/>
[s07]      <ds:DigestValue>h9kmx3rvDH75vKtNpi4NbeBGDnl=</ds:DigestValue>
[s08]    </ds:Reference>
[s09]    <ds:Reference URI="#SignedProperties"
        Type="http://uri.etsi.org/01903/v1.1.1#SignedProperties">
[s10]      <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[s11]      <ds:DigestValue> …..  </ds:DigestValue>
[s12]    </ds:Reference>
[s13]  </ds:SignedInfo>
[s14]  <ds:SignatureValue>…....</SignatureValue>
[s15]  <ds:KeyInfo> <ds:KeyValue> ….</ds:KeyValue></ds:KeyInfo>
[s16]  <ds:Object xmlns="http://uri.etsi.org/01903/v1.1.1#" xmlns:ds="
http://www.w3.org/2000/09/xmldsig#">
```

```
[s17]     <QualifyingProperties>
[s18]      <SignedProperties
Target="#SignatureWithSignedAndUnsignedProperties" Id="SignedProperties">
[s19]        <SignedSignatureProperties >
[s20]          <SigningTime>2000-11-18T12:10:00Z</SigningTime>
[s21]          <SigningCertificate>… … …</SigningCertificate >
[s22]          <SignaturePolicyIdentifier>…  …  …</ SignaturePolicyIdentifier
>
[s23]        </SignedSignatureProperties>
[s24]        <SignedDataObjectProperties>
[s25]          <DataObjectFormat>…  …  …</DataObjectFormat>
[s26]          <AllDataObjectsTimeStamp>… … … </AllDataObjectsTimeStamp>
[s27]        </SignedDataObjectProperties>
[s28]      </SignedProperties>
[s29]      <UnsignedProperties >
[s30]        <UnsignedSignatureProperties>
[s31]          <SignatureTimeStamp>…  …  …</SignatureTimeStamp>
[s32]          <CompleteCertificateRefs>…  …  …</CompleteCertificateRefs>
[s33]          <CompleteRevocationRefs>…  …  …</CompleteRevocationRefs>
[s34]          <SigAndRefsTimeStamp>… … …</SigAndRefsTimeStamp >
[s35]          <CertificateValues>…  …  …  …</CertificateValues>
[s36]          <RevocationValues>…  …  …</RevocationValues>
[s37]        </UnsignedSignatureProperties>
[s38]      </UnsignedProperties>
[s39]    </QualifyingProperties>
[s40]  </ds:Object>
[s41]</ds:Signature>
```

[s01] Beginning of the XML signature. The namespace by default is the namespace defined in XML-DIGSIG.

[s02]-[s13] The `ds:SignedInfo` element contains the information that is actually signed.

[s03] The `ds:CanonicalizationMethod` element indicates the algorithm used to get a canonical representation of the *SignedInfo* element before being signed.

[s04] The `ds:SignatureMethod` indicates the algorithms used to sign *SignedInfo*.

[s05] to [s16] `ds:Reference` elements contain the digest value and indication on the digest algorithm for each data object that has to be (indirectly) signed. Each one also has a reference to the corresponding data object. These elements also have the *Id* attribute that can be used to make individual references each one of them.

[s05-s08] The first `ds:Reference` element. Its URI attribute references the data object that has to be signed. `ds:DigestMethod` indicates the digest algorithm (sha1 in this case) and `ds:DigestValue` contains the digest value filtered in base 64.

[s09-s12] The second `ds:Reference` element. Its URI attribute points to the `SignedProperties` element(using the URI attribute) that contains the whole set of signed properties. `ds:DigestMethod` indicates the digest algorithm (sha1 in this case) and `ds:DigestValue` contains the digest value filtered in base. This means that the digest value of that `SignedProperties` is included in `SignedInfo` and in consequence signed when this element is signed. The `ds:Type` attribute indicates that this element is a reference to the SignatureProperties element as mandated in clause 6.3.1.

[s14] `ds:SignatureValue` contains the computed digital signature of `ds:SignedInfo` in base 64.

[s15] `ds:KeyInfo` contains cryptographic material to verify the signature.

[s16-s40] `ds:Object` contains three elements with the properties qualifying both the signature and the signed data object.

[s17-39] `QualifyingProperties` contains the full set of qualifying properties both signed (`SignedProperties`) and unsigned (Uns`ignedProperties`). The namespace by default is changed for this element and its contents to the one defined as namespace by default in the schema definition given in the present document in order not to have to qualify the whole set of elements. Additionally, as elements already defined in [5] are used in the definitions, its namespace is also defined (prefix `ds`).

[s18-s28] `SignedProperties` contains the whole set of qualifying properties that are signed grouped in two sequences. The first one (`SignedSignatureProperties`) contains all the signed properties that qualify the signature. The second one (`SignedDataObjectProperties`) contains all the signed properties that individually qualify each signed data object.

[s19-ss23] `SignedSignatureProperties` contains all the signed properties that qualify the signature (`SigningTime`, `SigningCertificate`, `SignaturePolicyIdentifier`).

[s20] `signingTime` contains the value of the signing instant when the signature has been computed.

[s21] `SigningCertificate` contains, as stated above, a restricted set of references to certificates to be used in verifying a signature.

[s24-27] `SignedDataObjectProperties` contains all the signed properties that individually qualify each signed data object (`AllDataObjectsTimeStamp, DataObjectFormat`).

[s25] `DataObjectFormat` identifies the format of the signed data object.

[s26] `AllDataObjectsTimeStamp` is a time-stamp issued for the signed data object.

[s29-38] `UnsignedProperties` contains the whole set of qualifying properties that are NOT signed

[s30-s37] `UnsignedSignatureProperties` the whole set of unsigned properties that qualify the signature.

[s31] `SignatureTimeStamp` contains a time-stamp for the signature itself.

[s32] `CompleteCertificateRefs` contains references to CA certificates in the certification path used to verify the signature.

[s33] `CompleteRevocationRefs` contains references to revocation information used to verify the signature.

[s34] `SigAndRefsTimeStamp` contains a time-stamp over the XAdES-C form of the electronic signature.

[s35] `CertificateValues` contains the values of the certificates referenced in `CompleteCertificateRefs`.

[s36] `RevocationValues` contains the revocation data used to validate the electronic signature.

NOTE: The tree shown in the example above does not explicitly show certain optional XML elements (like `Transforms` For a complete description of this tree see XML-Signature Core Syntax and Processing [5].

Below will follow the example of indirect incorporation of all the unsigned properties. In this example, the signed properties will be directly incorporated into the `ds:Signature` element as in the previous example. However, the unsigned properties will be separately stored in other place. To incorporate these properties use is made of the `QualifyingPropertiesReference` element pointing to the element containing them.

Below follows the contents of the XAdES itself.

```
[s01]<ds:Signature Id="SignatureWithSignedAndUnsignedProperties"
xmlns="http://www.w3.org/2000/09/xmldsig#>
[s02]  <ds:SignedInfo>
[s03]    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2000/WD-
xml-c14n-20000710"/>
[s04]    <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
[s05]    <ds:Reference URI="http://www.etsi.org/docToBeSigned"
Id="FirstSignedDocument">
[s06]      <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha"/>
[s07]      <ds:DigestValue>h9kmx3rvDH75vKtNpi4NbeBGDnl=</ds:DigestValue>
[s08]    </ds:Reference>
[s09]    <ds:Reference URI="#SignedProperties "
        Type=http://uri.etsi.org/01903/v1.1.1#SignedProperties>
[s10]      <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[s11]      <ds:DigestValue> …..  </ds:DigestValue>
```

```
[s12]    </ds:Reference>
[s13]  </ds:SignedInfo>
[s14]  <ds:SignatureValue>….....</SignatureValue>
[s15]  <ds:KeyInfo> <ds:KeyValue> ….</ds:KeyValue></ds:KeyInfo>
[s16]  <ds:Object xmlns="http://uri.etsi.org/01903/v1.1.1#" xmlns:ds="
http://www.w3.org/2000/09/xmldsig#">
[s17]     <QualifyingProperties>
[s18]      <SignedProperties
Target="#SignatureWithSignedAndUnsignedProperties" Id="SignedProperties">
[s19]        <SignedSignatureProperties >
[s20]          <SigningTime>2000-11-18T12:10:00Z</SigningTime>
[s21]          <SigningCertificate>… … …</SigningCertificate >
[s22]          <SignaturePolicyIdentifier>…  …  …</ SignaturePolicyIdentifier
>
[s23]        </SignedSignatureProperties>
[s24]        <SignedDataObjectProperties>
[s25]          <DataObjectFormat>…  …  …</DataObjectFormat>
[s26]          <AllDataObjectsTimeStamp>… … … </AllDataObjectsTimeStamp>
[s27]        </SignedDataObjectProperties>
[s28]      </SignedProperties>
[s28]    <QualifyingPropertiesReference>
[s29]      <Transforms> .. </Transforms>
[s30]      <URI>http://www.ac.upc.es/ETSI-XML/Indirect-
Incorporation/example1#QualifyingProperties</URI>
[s31]    </ QualifyingPropertiesReference>
[s32]    </QualifyingProperties>
[s33]  </ds:Object>
[s34]</ds:Signature>
```

[s1-s27] These lines are the same as in the first example. They show how the signed properties are directly incorporated.

[s28-s32] These lines show how to indirectly incorporate the unsigned properties stored in other place using the `QualifyingPropertiesReference` element.

[s29] The `Transforms` element contains the whole set of transformations to compute on the file where the unsigned properties are stored.

[s30] The `URI` element contains the URI pointing to the `QualifyingProperties` element that contains those qualifying properties that are being indirectly incorporated. In this case, it points to the file found in http://www.ac.upc.es/ETSI-XML/Indirect-Incorporation/example1, which contains this element.

This example ends showing that part of the file found in http://www.ac.upc.es/ETSI-XML/Indirect-Incorporation/example1 that contains the `QualifyingProperties` element referenced in the `QualifyingPropertiesReference.`

```
<!-- This is the part of the file found in http://www.ac.upc.es/ETSI-
XML/Indirect-Incorporation/example1 that contains the QualifyingProperties
element containing the unsigned properties that are indirectly incorporated
in the advanced electronic signature -->

[si]     <QualifyingProperties>
[si+1]      <UnsignedProperties >
[si+2]        <UnsignedSignatureProperties>
[si+3]          <SignatureTimeStamp>…  …  …</SignatureTimeStamp>
[si+4]          <CompleteCertificateRefs>…  …  …</CompleteCertificateRefs>
[si+5]          <CompleteRevocationRefs>…  …  …</CompleteRevocationRefs>
[si+6]          <SigAndRefsTimeStamp>… … …</SigAndRefsTimeStamp >
[si+7]          <CertificateValues>…  …  …  …</CertificateValues>
[si+8]          <RevocationValues>…  …  …</RevocationValues>
[si+9]        </UnsignedSignatureProperties>
[si+10]     </UnsignedProperties>
[si+11]  </QualifyingProperties>

<!-- Below would follow the rest of the file -->
```

In the example above the `QualifyingProperties` element is shown that is part of the file found in http://www.ac.upc.es/ETSI-XML/Indirect-Incorporation/example1 and that is pointed by the `URI` element in the `QualifyingPropertiesReference` in the advanced electronic signature.

History

Document history		
V1.1.1	February 2002	Publication