Draft EN 319 162-1V0.0.3 (2013-11)

Electronic Signatures and Infrastructures (ESI); Associated Signature Containers (ASiC); Part 1: Core Specification

STABLE DRAFT FOR PUBLIC REVIEW UNTIL 15 JANUARY 2014

Download the template for comments:

http://docbox.etsi.org/ESI/Open/Latest_Drafts/Template-for-comments.doc

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

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

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

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

Reference
DEN/ESI-0019162-1

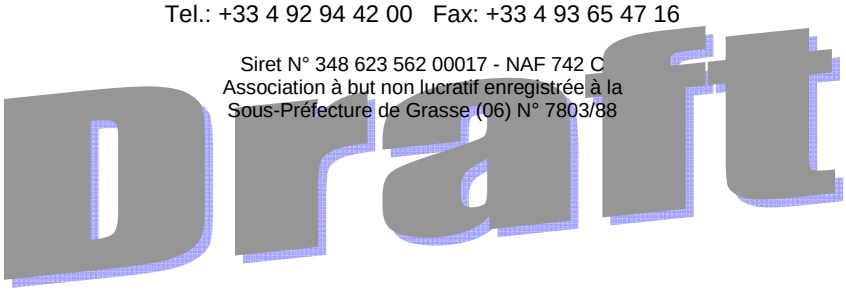
Keywords
ASiC, e-commerce, electronic signature, security

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



Important notice

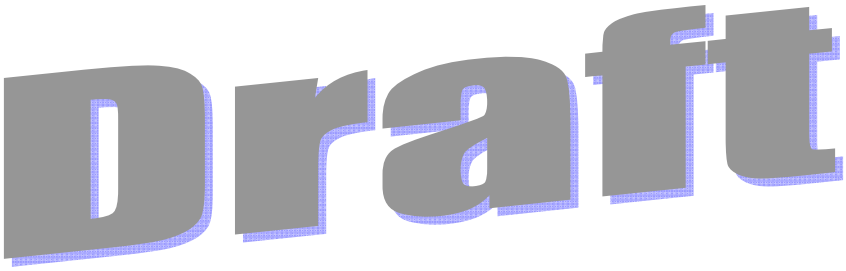
Individual copies of the present document can be downloaded from:
http://www.etsi.org

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

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification



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://ipr.etsi.org).

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 draft European Standard (EN) has been produced by ETSI Technical Committee Electronic Signatures and Infrastructures (ESI) and is now submitted for the Vote phase of the ETSI standards EN Approval Procedure.

The present document is part 1 of a multi-part deliverable.

The present document was previously published as TS 102 918 [i.7].

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

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 [i.3] 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". EN 319 122-1 [1] (CAdES) and EN 319 132-1 [2] (XAdES) define formats for electronic signatures in line with this definition. These formats include modes of use whereby the signature is detached from the data to which it is applied. The present document specifies the use of container structures for associating either detached CAdES signatures or detached XAdES signatures or time-stamp tokens, with one or more signed objects to which they apply.

1 Scope

The present document is part 1 of a multipart deliverable on Associated Signature Container (ASiC) and constitutes the core specification of ASiC specifying structures to bind together either detached advanced electronic signatures or time-stamp tokens, with a number of data objects (e.g. documents, XML structured data, spreadsheet, multimedia content) to which they apply into one single digital container based on ZIP [5] and supporting the following signature and time-stamp token formats:

- CAdES (EN 319 122-1 [1]);
- XAdES (EN 319 132-1 [2]);
- RFC 3161 [3] time-stamp tokens.

NOTE 1: No restriction is placed on the format of time-stamp tokens used within CAdES/XAdES.

A number of application environments use ZIP based container formats to package sets of data objects together with meta-information. ASiC is designed to operate with a range of such ZIP based application environments. Rather than enforcing a single packaging structure ASiC describes how these container formats can be used to associate advanced electronic signatures with any data objects in the container. In particular, the present documents aim is to work with implementations of OCF (OEBPS Container Format), ODF (Open Office), UCF or any similarly structured container format to also comply with one of the modes of use ASiC. It is also the aim of the present document to address use of "virtual dossiers" container formats such as required in some pan-European projects.

- Clause 4 provides a general introduction and background to ASiC.
- Clause 5 describes simple form which can be used for basic use cases where a single data object (e.g. a document), or a container has to be signed or time-stamped.
- Clause 6 describes extended form for use cases providing much greater flexibility in data objects protected by an individual signature.
- Clause 7 defines conformance requirements for ASiC implementations.
- Annex A specifies container metadata and referencing rules.
- Annex B gives examples of the use of the ASiC for particular applications.

New elements are defined in the present document to support additional features such as time-stamping and CAdES signing of multiple content and XAdES parallel signatures that may be used in other contexts.

The present document offers a basic support for TSTs, since it does not currently address the identification of the validation policy to be used for verifying a container that contains the TST.

2 References

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

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

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

2.1 Normative references

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

[1] ETSI EN 319 122-1: "Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signature Formats (CAdES); Part 1: Core Specification".

[2] ETSI EN 319 132-1: " Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signature Formats (XAdES); Part 1: Core Specification".

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

[4] IDPF: "EPUB Open Container Format (OCF)".

NOTE: Available at http://idpf.org/epub/30/spec/epub30-ocf.html.

[5] PKWARE ".ZIP Application Note".

NOTE: Available at http://www.pkware.com/support/zip-application-note.

[6] OASIS: "Open Document Format for Office Applications (OpenDocument) Version 1.2; Part 3: Packages" 29 September 2011. OASIS Standard.

[7] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

2.2 Informative references

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

[i.1] Adobe "Universal Container Format (UCF)".

NOTE: Available at http://livedocs.adobe.com/navigator/9/Navigator_SDK9_HTMLHelp/Appx_Packaging.6.1.html.

[i.2] ISO 15489-1: "Information and documentation - Records management - Part 1: General".

[i.3] Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures.

[i.4] OASIS "Guidelines to Writing Conformance Clauses", 4 September 2007.

NOTE: Available at http://docs.oasis-open.org/templates/TCHandbook/ConformanceGuidelines.html.

[i.5] W3C recommendation: "XML Signature Syntax and Processing".

[i.6] IETF RFC 4288: "Media Type Specifications and Registration Procedures".

[i.7] ETSI TS 102 918: "Electronic Signatures and Infrastructures (ESI); Associated Signature Containers (ASiC)".

[i.8] ETSI TS 101 861: "Electronic Signatures and Infrastructures (ESI); Time stamping profile".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in CAdES [1], XAdES [2] and the following apply:

conformance clause: statement of a specification that provides a high-level description of what is required for an implementation to conform by referring to other parts of the specification for details

NOTE 1: A Conformance Clause references one or more Normative Statements, directly or indirectly, and can refer to another Conformance Clause.

NOTE 2: Derived from [i.4].

container: file holding data objects with related manifest, metadata and associated signature(s), under a specified hierarchy

data object: any digital information to which Advanced Electronic Signature(s) and/or time-stamping are applicable

metadata: data describing context, content and structure of data objects and their management over time

NOTE: Refer to ISO 15489-1: 2001, definition 3.12 with modifications [i.2].

ZIP: data object conformant to [5]

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in CAdES [1], XAdES [2] and the following apply:

AdES Advanced Electronic Signature
ASiC Associated Signature Container
OCF OEBPS Container Format

NOTE: Refer to [4].

ODF Open Document Format

NOTE: Refer to [6].

OEBPS Open eBook Publication Structure
UCF Universal Container Format

NOTE: Refer to [i.1].

4 Introduction to Associated Signature Containers (informative)

4.1 Requirements addressed by Associated Signature Containers

When signing data, the resultant signature needs to be associated with the data to which it applies. This can be achieved either by creating a data set which combines the signature and the data that was signed (e.g. by enveloping the data with the signature or including a signature element in the data set) or placing the (detached) signature in a separate resource

and have some external means for associating the signature with the data to which it applies. While there are some advantages to the use of detached signatures, most significantly their non-modification of the original data objects, there remains a risk that the signature becomes separated from the data to which it applies and so losing the association. Therefore, many application systems have developed their own technique for combining a detached signature with the signed object in some form of container so that they can be more easily distributed and guarantee that the correct signature and any relevant metadata is used when verifying. The same requirements apply to associate a time-stamp token to its related data.

The present document defines a standardized use of container forms to establish a common way for associating data objects with advanced signatures or time-stamp tokens. Using a common container form and associated information will enable a greater amount of interchange and interoperability among various signing and verification services.

Whilst ZIP [5] provides a basic container structure that can associate data objects and the signature(s) that apply to them, there is a recognized need for additional structure and metadata about the association, for example to link a particular signature with the data object to which it is applied. Other formats have already been specified for the use of ZIP based structures to bind together a number of data objects with related metadata. This includes OCF [4] (OEBPS Container Format) which was originally designed for use by eBooks but has been adopted as the basis for other containers including that used by ODF [6] (Open Document Format - Open Office) and UCF [i.1] (Universal Container Format by Adobe Systems). The present document builds on this work specifically addressing the requirements of associating an advanced electronic signature with any type of data, independent of the needs of any particular document or data type. The present document can be used for any type of virtual dossier types that collects together electronic documents including those supported by OCF, ODF and UCF.

4.2 Main features of Associated Signature Containers

4.2.1 Basic container structure

The ASiC is a data container holding a set of data objects and associated signatures using the ZIP [5] format. The ZIP format was chosen as it is used by many popular container formats and it is natively supported by most operating systems.

Any ASiC container has an internal structure including:

- a root folder, for all the container content possibly including folders reflecting the content structure;
- a META-INF folder, in the root folder, for metadata about the content, including associated signatures.

The detached signatures are applied in such a way that the integrity of the data is not broken when the signed objects are extracted from the ZIP container. Hence, the signatures used in ASiC can be verified against the protected data objects when outside the container structure (for example when placed in local storage).

4.2.2 Container types

The present document defines two types of containers.

The first type (ASiC-S) is a simple container to associate one or more signatures with a single data object. The signatures are carried in a single signature structure which may either be:

- a single CAdES signature which may contain parallel signatures; each of them can further be individually counter-signed; or
- multiple XAdES signatures using the new structure defined in clause A.4; each of them can further be individually counter-signed; or
- a single TST.

The container contains only the data object and the signature or the time-stamp token that applies to it, with an optional mimetype. Parallel signatures are supported and it is possible to add at a later time additional signatures to the same data object. The signed data object can itself be a container nested within the ASiC container.

The second is an extended container (ASiC-E) that contains multiple data objects. Each data object may be signed by one or more signature structures carried in the container.

Each signature structure may be either:

- a single CAdES signature which may contain parallel signatures; each of them can further be individually counter-signed; or
- one or more XAdES signatures using the new structure defined in A.4; each of them can further be individually counter-signed; or
- a single TST.

This second type of container is compatible with OCF, UCF and ODF formats.

Data objects are signed, together with some metadata and each signature is associated with all or some of the data objects in the container. It is possible to add signatures and data objects to an Extended ASiC and the additional signatures can apply to the same or different set of data objects, without invalidating previously applied signatures. Later signatures may sign signatures applied previously.

ASiC is based on CAdES [1] or XAdES [2] Advances Electronic Signatures or on time-stamp tokens conformant to RFC 3161 [3] that may be profiled as specified in TS 101 861 [i.8].

NOTE 1: Future versions of the present document may support additional formats.

NOTE 2: CAdES and XAdES Archive Time-stamp attributes do not protect data objects referenced using ASiCManifest and ds:Manifest. For long term validation see clause 5.4 for ASiC-S and 6.5 for ASiC-E.

All ASiC container types support parallel signatures.

4.3 Compliance with external specifications

The extended ASiC type can be used in a way which is compatible with OCF, UCF and ODF. These container specifications define a structure with metadata about its content.

The following are examples of metadata elements from these container specifications that can be used with ASiC:

- "mimetype" containing the mime type to identify the container type, conforming to relevant clauses in the present document (supported by OCF, UCF and ODF).
- "META-INF/container.xml" metadata about container content allowing to specify one or more root data objects to specify how to start processing the container (mandatory in OCF, not allowed by ODF and not required by UCF).
- "META-INF/manifest.xml" additional metadata about container content - (allowed by OCF and UCF and mandatory in ODF).
- "META-INF/metadata.xml" that can contain user defined metadata associated with data objects (allowed by OCF and UCF not allowed by ODF).
- "META-INF/*signatures*.xml" that provide support to include one or more XML signatures, each one of them applicable to some or all the data objects present in the ASiC container. Electronic signature is allowed by OCF, ODF and UCF but with slightly different syntaxes, all supported by the present document with specific provisions in clause 6.

These and other elements of the OCF, UCF and ODF formats may be combined within ASiC as appropriate; see annex C for an informative cross reference of possible content in different containers.

5 Associated Signature Simple form

This clause defines the form of the Simple Associated Signature Containers (ASiC-S) that associates a single data object with one or more detached signature(s) or a single time-stamp token that apply to it. This form supports the use of CAdES or XAdES signatures as well as time-stamp tokens. The time-stamp token is a binary representation of TimeStampToken as defined in RFC 3161 [3].

ASiC-S is also applicable when the signed data object is itself a container, for example ZIP, OCF, ODF, UCF, or another ASiC. In this case it associates the inner container with one or more signatures or a time-stamp token that applies to it. Examples of the use of ASiC-S are given in clause B.1.

5.1 General Requirements for ASiC-S

The ZIP format [5] with the structure specified in clause 5.2.2 shall be used to bind the contained objects into a single container.

Implementations may support ASiC-S for a single form type (i.e. with at least one of CAdES or XAdES or RFC 3161 [3] time-stamp token).

NOTE: Future versions of the present document may allow use of other signature formats.

5.2 Detailed format for ASiC-S

5.2.1 Media type identification

1) File extension: ".asics" should be used (".scs" should be used in case of operating systems and/or file systems not allowing more than 3 characters file extensions). In the case that the container content is to be handled manually, the ".zip" extension may be used.

2) Mime type: implementers may use "application/vnd.etsi.asic-s+zip" to identify this format or may maintain the original mimetype of the signed data object.

3) The archive level comment field in the ZIP header may be used to identify the type of the data object within the container. If this field is present, it should be set with "mimetype=" followed by the mime type defined above and shall have the same value of the "mimetype" specified in clause 5.2.2 item 1 if present.

NOTE: The mime type can include parameters, e.g. a "charset" parameter can be used to indicate the charset of the body text like for "text/plain" mime type (see [i.6] clause 4.2.1).

5.2.2 Contents of the container

This container is built using the following internal structure:

1) An optional "mimetype", defined in clause A.1, containing the mime type as defined in clause 5.2.1 item 2. If the container file extension does not imply use of ASiC then the "mimetype" shall be present and the first option in clause 5.2.1 item 2 shall apply.

2) The signed data object in the root level of the container. It shall be the only data object present at the container root level besides the eventual "mimetype" specified in item 1) above.

NOTE 1: The signed object can be itself a container, for example in one of the following formats: ZIP, ASiC, OCF, ODF or UCF.

3) The META-INF folder containing one of:

   a) "timestamp.tst" containing a binary TimeStampToken as defined in RFC 3161 [3], calculated over the entire binary content of the data object specified in item 2; or

   b) "signature.p7s" containing the detached signature in CAdES format of the data object specified in item 2. Multiple parallel signatures and countersignatures are allowed, provided they are all collected in the same CAdES structure; or

   c) "signatures.xml" containing the root element `<asic:XAdESSignatures>` as specified in clause A.5, containing one or more detached ds:Signature elements signing the whole data object specified in item 2 and conformant to XAdES. For ASiC-S `<ds:Reference>` shall be used to reference the data object in the container and the rules specified in clause A.6 shall apply. In case URI is not present in `<ds:Reference>` element then a reference to the signed content is implied. Any canonicalization computed on descendant

elements of a `<ds:Signature>` shall be performed keeping this <ds:Signature> element as a child of `<asic:XAdESSignatures>` (without detaching it).

NOTE 2: In the case of use of implied reference the party verifying the signature is aware of the application context and the expected relation between the signed data object and the signature. Use of implied reference gives greater flexibility for the application's use of ASiC in positioning the signature relative to the data. Use of relative references requires the relative positioning to be maintained when data is extracted from the container if signatures are still to be verifiable.

NOTE 3: Use of exclusive canonicalization is also allowed. In this case the canonicalization result does not include the ancestor's context (`<asic:XAdESSignatures>` element in this case).

Other application specific information may be added in further metadata objects contained within the META-INFO folder.

Figures 1 to 5 illustrate examples for the content of the ASiC-S container.

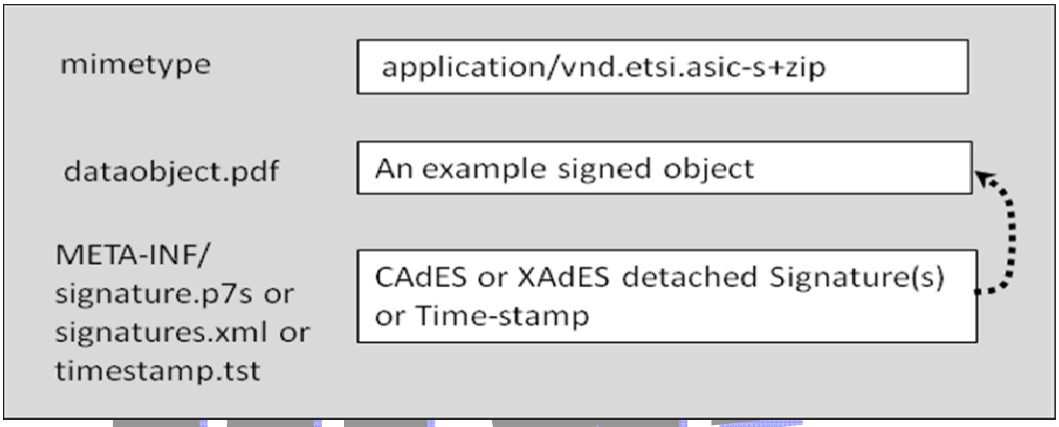


Figure 1: ASiC-S structure applied to a plain data object

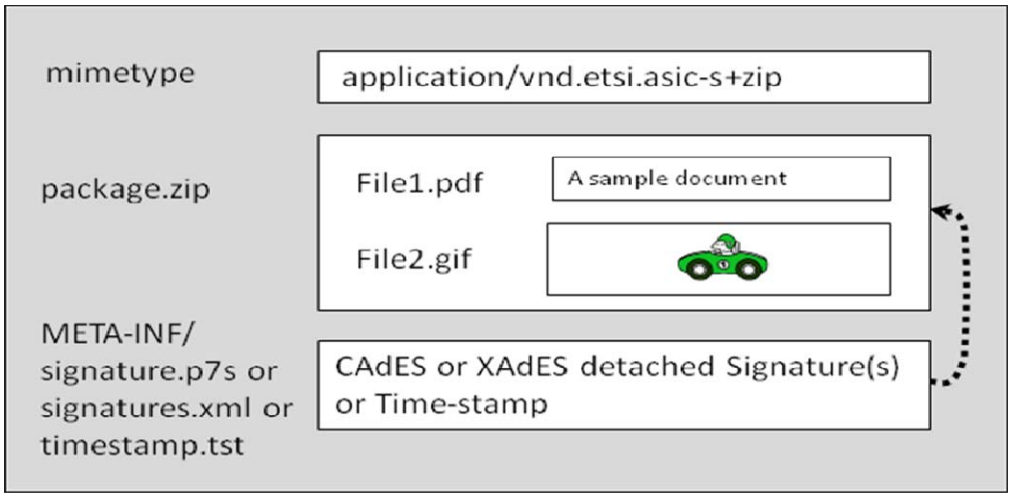


Figure 2: ASiC-S structure applied to a nested container

5.3 Mime type correlation check

If the "mimetype" is present conformant implementations shall check that it is consistent with provisions in clause 5.2.1 item 2 and clause 5.2.2 item 1.

5.4 Attributes for long term validation of ASiC-S

Long term validation of ASiC-S is achieved for the different container types as follows:

1) When it is necesary to protect for long term an ASiC-S container with a CAdES signature (including a single signature or a number of parallel signatures) the long term attributes of CAdES shall be used according to its specification [1].

2) When it is necesary to protect for long term an ASiC-S container with one or more XAdES signatures, the long term attributes of XAdES shall be used according to the its specification [2] to all the signatures present in signatures.xml.

Figure 3 shows an ASiC-S container with a CAdES or a XAdES signature with long term attributes.

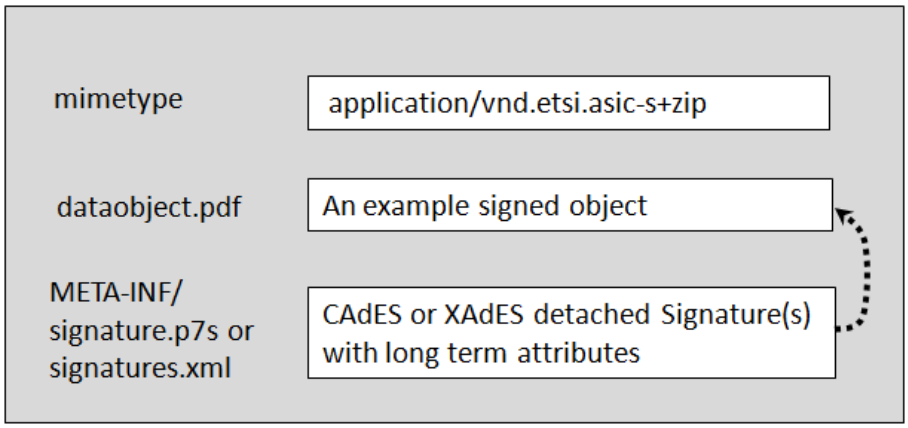


Figure 3: ASiC-S with XAdES or CAdES long term attributes

3) When it is necesary to protect for long term an ASiC-S with Time-stamp token a META-INF/ASiCArchiveManifest.xml and a META-/INF/*timestamp*.tst that applies to it shall be added to the container.

ASiCArchiveManifest.xml shall have the same syntax of ASiCManifest.xml and reference the time-stamped object and the associated Timestamp.tst that applies to it.

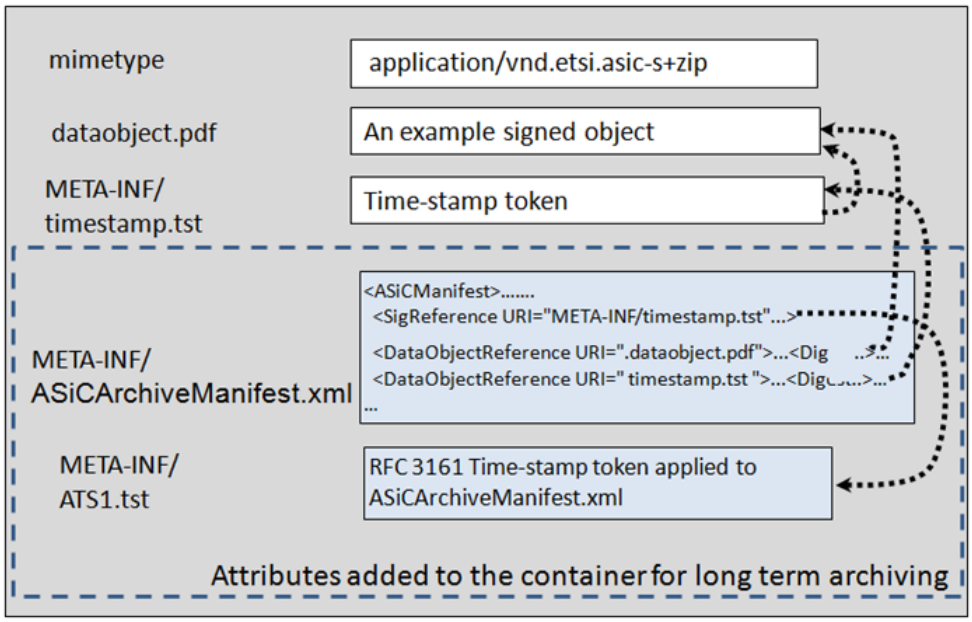


Figure 4: ASiC-S with time-stamp token and long term attributes

Figure 4 shows an ASiC-S container with a time-stamp token with long term attributes.

If a META-INF/ASiCArchiveManifest.xml is already present in the container it shall be renamed to any valid META-INF/*ASiCArchiveManifest*.xml not already present in the container.

A META-INF/ASiCArchiveManifest.xml and a META-INF/*timestamp*.tst that applies to it shall be added to the container referencing the time-stamped object, the associated Timestamp.tst that applies to it, any META-INF/*ASiCArchiveManifest*.xml and the associated META-/INF/*timestamp*.tst that applies to it. The renamed ASiCArchiveManifest shall be set with Rootfile true.

Any META-INF/*timestamp*.tst shall contain all the validation information needed to verify the time-stamp token before a META-INF/*ASiCArchiveManifest*.xml refers to it. In case additional validation data is needed to verify one or more META-INF/* timestamp*.tst already present in the container and referenced by a META-INF/*ASiCArchiveManifest*.xml the validation data shall be added to the last META-INF/* timestamp*.tst (i.e. the one that protects META-INF/ASiCArchiveManifest.xml.

Figure 5shows an ASiC-S container with a time-stamp token with long term attributes.

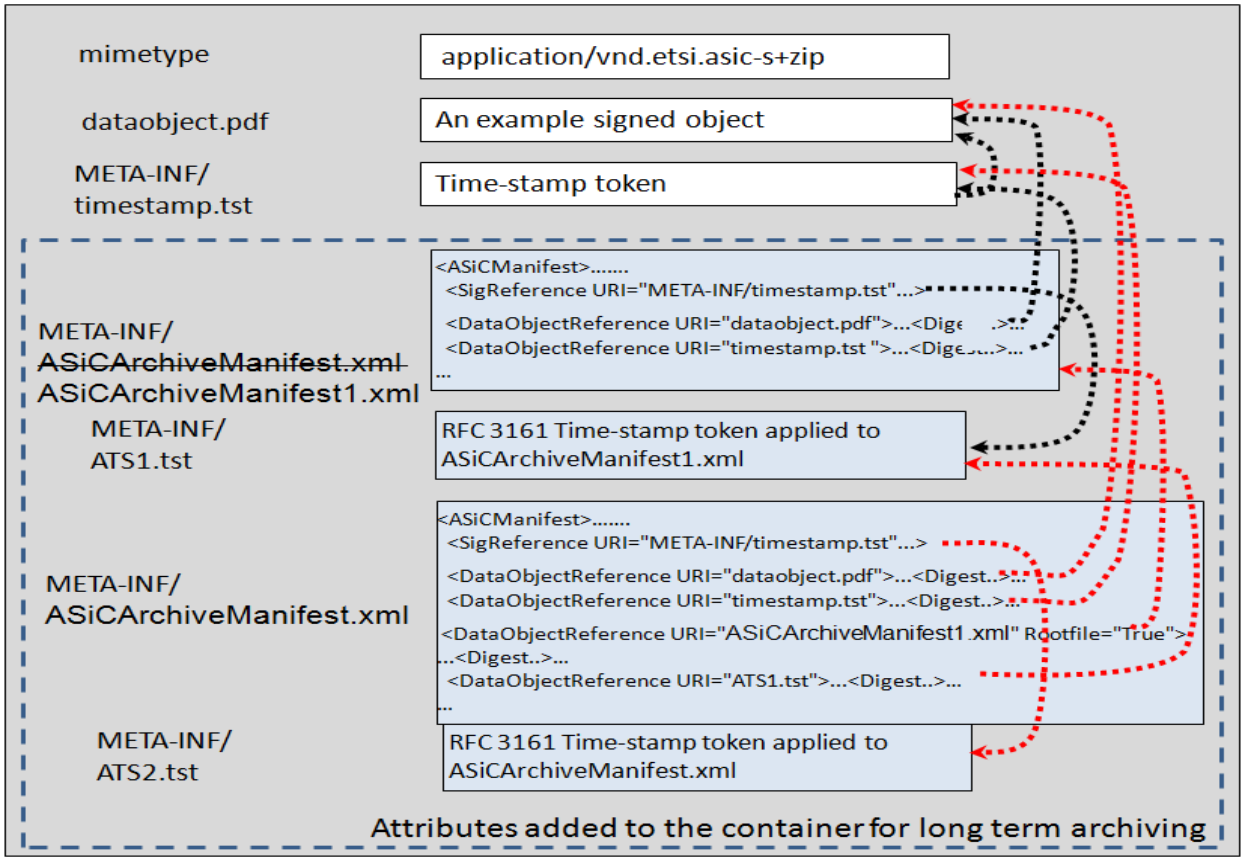


Figure 5: adding an ASiCArchiveManifest.xml to extend the ASiC-S with time-stamp token validity

6 Associated Signature Extended form

The Extended ASiC container types support one or more signatures and time-stamp tokens each applicable to its own set of one or more data objects. Each data object can have associated additional information and metadata that can also be protected by any of the signature(s) present in the container. The container packages all the mentioned elements. The container can be designed to prevent any further modification or allowing that additional data objects and signatures can be included at a later time to the container without breaking the previous signatures.

Two Extended ASiC structures are defined:

1) Associated Signature Container Extended form using XAdES: the data objects are associated with signatures in XAdES format.

2) Associated Signature Container Extended form using CAdES or Time-Stamp Tokens: the data objects are associated with signatures in CAdES format or with Time Stamp Tokens.

All ASiC types allow container nesting (with inner containers being themselves ASiC or any type of container) allowing arbitrary complex hierarchies to be represented.

The clauses in this section specify Extended ASiC structures. Clauses 7.2.1, 7.2.2 and 7.2.3 refer to specific clauses and items in this section that implementations claiming conformance to each ASiC form shall conform to. When using ASiC with an external specification, applicable clauses and items that implementations shall conform to are specified in clause 7.2.4.

6.1 General Requirement of ASiC-E

The ZIP format [5] shall be used to bind the contained objects into a single container.

Implementations may support just the ASiC-E for a single form type (i.e. with at least one of CAdES, time-stamp token or XAdES); see clauses 7.2.1, 7.2.2 and 7.2.3 for more details.

6.2 Detailed format for ASiC-E with XAdES

The structure specified in clause 6.2.2 shall be used to build the container.

One or more signatures can be present in this form type. Each signature can contain an arbitrary number of ds:Signature elements, each signing an arbitrary set of data objects within the container that shall be referenced according to rules defined in clauses A.6 and 6.2.4.

6.2.1 Media type identification

4) File extension: ".asice" should be used (".sce" is allowed for operating systems and/or file systems not allowing more than 3 characters file extensions).

5) Mime type: "application/vnd.etsi.asic-e+zip" mime type should be used to identify the format of this container.

6) The archive level comment field in the ZIP header may be used to identify the type of content within the container. If this field is present, it should be set with "mimetype=" followed by the mime type defined above and shall have the same value of the "mimetype" specified in clause 6.2.2 item 1 if present.

6.2.2 Contents of Container

Signatures associated to data objects are based on XAdES signatures. Clauses A.6 and 6.2.4 specify the rules on referencing signed data objects.

This container is in zip format with the content and internal structure defined as follows:

1) An optional "mimetype", defined in clause A.1, containing the mime type as defined in clause 6.2.1, item 2. If the container file extension does not imply use of a supported container format then the "mimetype" shall be present.

2) One or more "*signatures*.xml" in a path beginning with META-INF/ shall be present containing one or more XAdES signatures conforming to [2] as specified in the following item. Signed data objects may either be directly referenced by each signature with a set of `<ds:Reference>` elements or may be indirectly referenced using a `<ds:Manifest>` object that is pointed by a `<ds:Reference>`, following the rules specified in clause 6.2.4.

3) The root element of each "*signatures*.xml" content shall be either:

   d) `<asic:XAdESSignatures>` as specified in clause A.5, the recommended format; or

   e) `<document-signatures>` as specified in OASIS Open Document Format [6]; or

   f) `<signatures>` as specified in OEBPS Container Format (OCF) [4]; or

   g) any other element in any namespace including the ds:Signature element itself.

All the root elements inside the signature files in a given container should be the same. Any canonicalization computed on descendant elements of a `<ds:Signature>` shall be performed keeping this `<ds:Signature>` element as a child of the root element, without detaching it.

NOTE 1: As specified in clause A.4 and in OCF [4] and ODF [6], in either case the child elements of the root element are one or more `<ds:Signature>` sibling elements as specified in W3C recommendation: "XML Signature Syntax and Processing" [i.5].

NOTE 2: Use of exclusive canonicalization is allowed: in this case the canonicalization result does not include the ancestor's context (`<asic:XAdESSignatures>` element in this case).

NOTE 3: Item 3)d) allows migrating existing, legacy, detached and/or enveloped signatures that contains explicit or implicit inclusive canonicalization into an ASiC-E container.

4) Other application specific information may be added in further files contained within the META-INF directory, such as:

a) "META-INF/container.xml" if present shall be well formed XML conformant to OEBPS Container Format (OCF) [4] specifications. It shall identify the MIME type and full path of all the root data objects in the container, as specified in OCF.

b) "META-INF/manifest.xml" if present shall be well formed XML conformant to OASIS Open Document Format [6] specifications.

NOTE 4: according to ODF [6] specifications, inclusion of reference to other META-INF information, such as *signatures*.xml, in manifest.xml is optional. In this way it is possible to protect the container's content signing manifest.xml while allowing to add later signatures.

c) "META-INF/metadata.xml" has a user defined content. If present, its content shall be well formed XML conformant to OEBPS Container Format (OCF) [4] specifications.

6.2.3 ASiC-E with XAdES example (informative)

In figure 3 is represented a typical structure for this container where the XMLDSig [i-5] element ds:Reference is used directly to reference the signed objects. Implementers are advised that use of ds:Manifest requires special attention and specific requirements are given in clause 6.2.4.

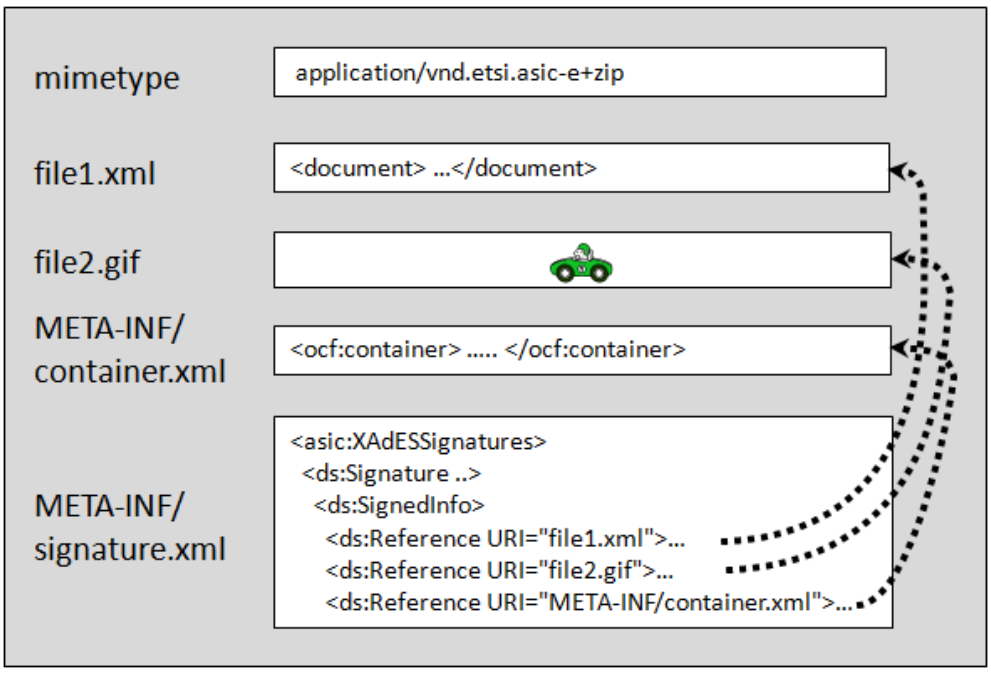


Figure 6: ASiC-E with XAdES and direct ds:reference usage

6.2.4 XAdES use in ASiC-E with XAdES

For ASiC-E used with XAdES the rules specified in clause A.6 shall apply.

To reference the signed data objects ds:Reference should be used in preference to ds:Manifest.

In the case that ds:Manifest element is used the following restrictions shall apply:

1) The ds:Manifest containing ds:Reference elements referencing the signed data objects shall be signed (i.e. shall be referenced within ds:SignedInfo element and its contents contribute to the ds:SignatureValue content).

2) The ds:Manifest elements shall not reference other ds:Manifest elements within a ds:Signature (i.e. direct chaining of ds:Manifest is not allowed).

3) Applications claiming compliance with the present document, shall raise a warning whenever a digest value mismatch is detected within any ds:Manifest's ds:Reference child (i.e. the digest computed over the referenced data object and the ds:DigestValue within this ds:Reference do not match), even if the cryptographic verification of the ds:SignedInfo succeeds (i.e. if the signature value computed by the verifying application actually matches ds:SignatureValue's content).

4) For referencing the ds:Manifest element from the ds:Reference element in the corresponding signature an Id attribute should be used.

The process for the verification of the ds:Manifest is outside the scope of the present document.

6.3 Detailed format for ASiC-E with CAdES and Time Stamp Tokens

The ASiC-E form can be used to apply CAdES signatures and time-stamp tokens to a set of data objects.

Each CAdES signature allows presence of parallel signatures and countersignatures.

6.3.1 Media type identification

1) File extension: ".asice" should be used (".sce" is allowed for operating systems and/or file systems not allowing more than 3 characters file extensions).

2) Mime type "application/vnd.etsi.asic-e+zip" mime type should be used to identify the format of this container.

3) The archive level comment field in the ZIP header may be used to identify the type of content within the container. If this field is present, it should be set with "mimetype=" followed by the mime type defined above and shall have the same value of the "mimetype" specified in clause 6.3.2 item 1 if present.

6.3.2 Contents of Container

This container is in zip format with the content and internal structure defined as follows:

1) An optional "mimetype" containing the mime type defined in clause 6.3.1, see clause A.1 for requirements on encoding mimetype.

2) Any number of data objects that can be digitally signed arbitrarily structured in folders.

3) At least one "META-INF/ASiCManifest*.xml", with the content specified in clause A.4, shall be present.

4) For each "META-INF/ASiCManifest*.xml" a time-stamp token or a signature that applies to it shall be present named as follows:

   - a) "META-INF/*signature*.p7s" containing a CAdES signature conforming to [1]; or

   - b) "META-INF/*timestamp*.tst" containing a binary TimeStampToken as defined in RFC 3161 [3].

Implementations claiming conformance to ASiC-E with CAdES, according to clause 7.2.2, and/or time-stamp token, according to clause 7.2.3, shall:

1) verify the signature or time-stamp, as applicable;

2) verify that the content signed or time-stamped conforms to clause A.4;

3) raise an error whenever a digest value mismatch is detected within any ds:DigestValue in asic:DataObjectReference and the digest computed over the referenced data object, even if the cryptographic verification of the signature or time-stamp token succeeds.

NOTE: Implementors of ASiC are warned that CAdES compliant verifiers are not aware of ASiC specific rules covers only step 1) above and do not guarantee full compliance to ASiC if also steps 2) and 3) are implemented.

Figure 4 shows an example for the content of the ASiC-E where a CAdES signature is applied to a set of data objects.

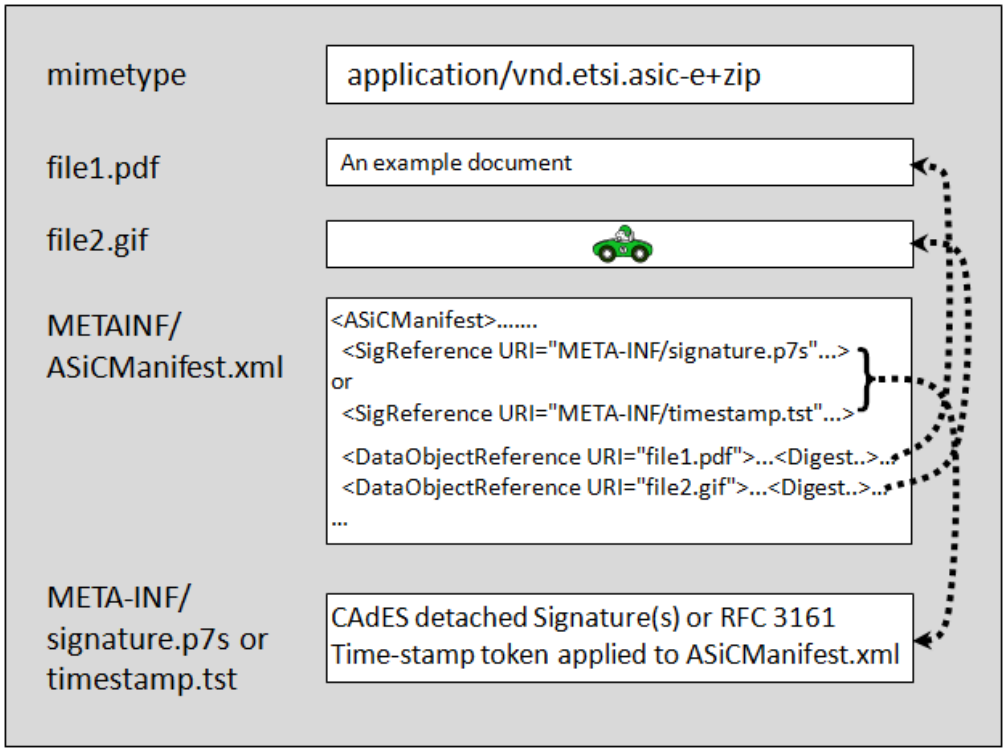


Figure 7: ASiC-E with CAdES signature

This container format allows the application of different CAdES signatures and/or time-stamp tokens to different set of data objects, as shown in figure 5.

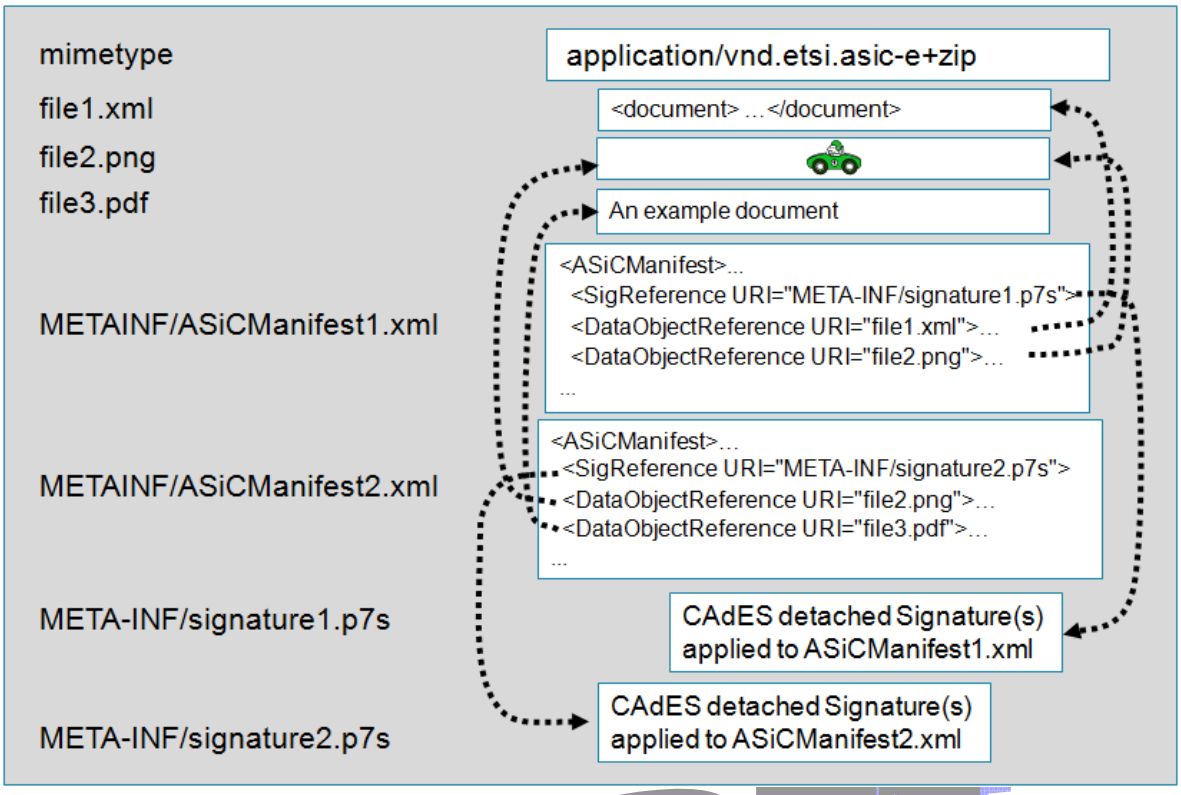


Figure 8: ASiC-E with CAdES containing different signatures

6.4 Mime type correlation check

If the "mimetype" is present conformant implementations shall check that it is consistent with clause 6.2.1 item 2 and clause 6.2.2 item 1. Also, if the META-INF includes manifest objects containing the mimetype of the object referenced (e.g. ASiCManifest*.xml) implementations shall check that this is coherent with the object referenced.

6.5 Attributes for long term validation of ASiC-E

Long term validation of ASiC-E is achieved for the different container types as follows:

1) When it is necessary to protect for long term an ASiC-E container with one or more XAdES signatures, the long term attributes shall be used according to the XAdES specification [2] in all the signatures present in signatures.xml.

2) When it is necesary to protect for long term an ASiC-E with with one or more CAdES signatures or a Time-stamp token a META-INF/ASiCArchiveManifest.xml and a META-INF/*timestamp*.tst that applies to it shall be added to the container.

   ASiCArchiveManifest.xml shall have the same syntax of ASiCManifest.xml and reference at least all the data objects referenced by at least one of the META-INF/*ASiCManifest*.xml already present in the container and all the related signatures and time-stamp tokesn.

   If a META-INF/ASiCArchiveManifest.xml is already present in the container it shall be renamed to any valid META-INF/*ASiCArchiveManifest*.xml not already present in the container. A META-INF/ASiCArchiveManifest.xml and a META-/INF/*timestamp*.tst that shall be added to the container referencing, in addition to what specified for ASiCArchiveManifest.xml, all the META-INF/*ASiCArchiveManifest*.xml and the associated META-/INF/*timestamp*.tst that applies to it. The renamed ASiCArchiveManifest shall be set with Rootfile true.

   Any META-INF/*timestamp*.tst shall contain all the validation information needed to verify the time-stamp token before a META-INF/*ASiCArchiveManifest*.xml refers to it. In case additional validation data is needed to verify one or more META-INF/* timestamp*.tst already present in the container and referenced by a

META-INF/*ASiCArchiveManifest*.xml the validation data shall be added to the last META-INF/* timestamp*.tst (i.e. the one that protects META-INF/ASiCArchiveManifest.xml.

7 Conformance requirements

The present clause specifies conformance requirements for the generation and/or verification of the Associated Signature Containers specified in the present document.

An implementation can claim conformance to ASiC if it supports at least one of the following Conformance Clauses. Conformance to any set of the following Conformance Clauses may be claimed and explicit reference to those supported may be used to profile specific interoperable implementations.

7.1 ASiC-S conformance

Implementations claiming conformance to ASiC-S shall meet the requirements of the present document for generating and/or verifying Simple Associated Signature Type (ASiC-S) as specified in one or more of the following clauses.

7.1.1 ASiC-S CAdES Conformance Clause

Implementations claiming conformance to this clause shall meet the requirements specified in clauses 5.1, 5.2.1 and 5.2.2, items 1, 2 and 3b. A verifying implementation shall support clause 5.3.

7.1.1.1 ASiC-S CAdES long term Conformance Clause

Implementations claiming conformance to this clause shall meet the requirements specified in clause 7.1.1 and 5.4 point 1).

7.1.2 ASiC-S XAdES Conformance Clause

Applications claiming conformance to this clause shall meet the requirements specified in clauses 5.1, 5.2.1 and 5.2.2, items 1, 2 and 3c. A verifying implementation shall support clause 5.3.

7.1.2.1 ASiC-S XAdES long term Conformance Clause

Implementations claiming conformance to this clause shall meet the requirements specified in clause 7.1.2 and 5.4 point 2).

7.1.3 ASiC-S Time-stamp token Conformance Clause

Implementations claiming conformance to this clause shall meet the requirements specified in clauses 5.1, 5.2.1 and 5.2.2, items 1, 2 and 3a. A verifying implementation shall support clause 5.3.

7.1.3.1 ASiC-S Time-stamp token long term Conformance Clause

Implementations claiming conformance to this clause shall meet the requirements specified in clause 7.1.3 and 5.4 point 3).

7.2 ASiC-E conformance

Implementations claiming conformance to ASiC-E shall meet the requirements specified in the present document for generating and/or verifying Extended Associated Signature Type (ASiC-E) as follows.

7.2.1 ASiC-E XAdES Conformance Clause

Implementations claiming conformance to this clause shall meet the requirements specified in clauses 6.1, 6.2.1 and 6.2.2. Furthermore, the implementation shall support:

At least one of the media type identification specified in clause 6.2.1 and both items 1 and 2 should be supported.

At least one of the content specified in clause 6.2.2 item 4a or 4b.

A verifying implementation, shall support clause 6.4.

7.2.1.1 ASiC-E XAdES long term Conformance Clause

Implementations claiming conformance to this clause shall meet the requirements specified in clause 7.2.1 and 6.5 point 1).

7.2.2 ASiC-E CAdES Conformance Clause

Implementations claiming conformance to this clause shall meet the requirements specified in clauses 6.1, 6.3.1 and 6.3.2 (excluded item 4b unless conformance to ASiC-E Time-stamp is also claimed). If a verifying implementation, it shall support clause 6.4. Furthermore, the implementation shall support at least one of the media type identification specified in clause 6.3.1 items 1 and 2 and should support both.

7.2.2.1 ASiC-E CAdES long term Conformance Clause

Implementations claiming conformance to this clause shall meet the requirements specified in clause 7.2.2 and 6.5 point 2).

7.2.3 ASiC-E Time-stamp token Conformance Clause

Implementations claiming conformance to this clause shall meet the requirements specified in clauses 6.1, 6.3.1 and 6.3.2 (excluded item 4a unless conformance to ASiC-E CAdES is also claimed). If a verifying implementation, it shall support clause 6.4. Furthermore, the implementation shall support at least one of the media type identification specified in clause 6.3.1 items 1 and 2 and should support both.

7.2.3.1 ASiC-E Time-stamp token long term Conformance Clause

Implementations claiming conformance to this clause shall meet the requirements specified in clause 7.2.3 and 6.5 point 2).

7.2.4 ASiC-E other container Conformance Clause

Implementations claiming conformance to this clause are expected to claim conformance also to an external container technical specification or standard. In addition, such implementations shall support the clauses 6.1 and 6.2.2 items 2 and 3, that shall have precedence over the external specification requirements.

Additional requirements specified in clause 6 apply if not in contradiction with the external specification.

7.2.4.1 ASiC-E other container long term Conformance Clause

Implementations claiming conformance to this clause shall meet the requirements specified in clause 7.2.4 and 6.5 point 1).

Annex A (normative): ASiC metadata specification and data naming and referencing

This annex contains the definition of ASiC specific content or additional specification for metadata already defined in other container formats and specify data objects and metadata naming and referencing rules.

A.1 Mimetype

The "mimetype" object, when stored in a ZIP, file can be used to support operating systems that rely on some content in specific positions in a file (the so called "magic number" as described in RFC 4288 [i.6] in order to select the specific application that can load and elaborate the file content. The following restrictions apply to the mimetype to support this feature:

- it has to be the first in the archive;
- it cannot contain "Extra fields" (i.e. extra field length at offset 28 shall be zero);
- it cannot be compressed (i.e. compression method at offset 8 shall be zero);
- the first 4 octets shall have the hex values: "50 4B 03 04".

An application can ascertain if this feature is used by checking if the string "mimetype" is found starting at offset 30. In this case it can be assumed that a string representing the container mime type is present starting at offset 38; the length of this string is contained in the 4 octets starting at offset 18.

All multi-octets values are little-endian.

The "mimetype" shall NOT be compressed or encrypted inside the ZIP file.

A.2 MIME registrations

The following MIME-Types and file-extensions are used in the present document:

NOTE: each MIME-Type is registered with IANA, additional information is available in the list of Directories of Content Types and Subtypes that can be found here: http://www.iana.org/assignments/media-types/application/.

MIME media type name:	Application
MIME subtype name:	vnd.etsi.asic-s+zip
Required parameters:	none
encoding considerations:	binary
File extension:	asics or scs

MIME media type name:	Application
MIME subtype name:	vnd.etsi.asic-e+zip
Required parameters:	none
encoding considerations:	binary
File extension:	asice or sce

MIME media type name:	Application
MIME subtype name:	vnd.etsi.timestamp-token
Required parameters:	none
encoding considerations:	binary
File extension:	tst

Security considerations:	The data objects carried in ASiC container may contain malicious code and hence unless the source is trusted the usual protection against malware and viruses should be applied. The integrity of the data is guaranteed by the electronic signature when present or should be provided externally if only a time-stamp token is applied to the data. Privacy can be guaranteed through the use of ZIP encryption features or externally. External integrity and privacy protection can be obtained e.g. through the use of SSL/TLS or S/MIME.
Published specification:	The ASiC formats as defined in the present document.

A.3 ASiC XML Schema

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

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  targetNamespace="http://uri.etsi.org/02918/v1.2.1#"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns="http://uri.etsi.org/02918/v1.2.1#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
   <xsd:import
     namespace="http://www.w3.org/2000/09/xmldsig#"
     schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd"/>
```

This XML Schema shown in the present annex is held in the file en_31916201v010101p0.zip attached to the present document as a normative part. In any case of difference in contents between the present document and the attached file, the reference version is the present document.

The hash values calculated on the schema file are:

MD-5: 9f:20:79:aa:0d:f9:48:1e:5a:75:c8:b8:34:27:be:b6

SHA-1: 91:38:b3:d3:94:d8:e3:dc:16:7b:03:fe:99:f5:4d:1c:af:9b:4b:65

SHA-256: 34:c0:84:a8:2c:03:16:a5:94:b7:c8:90:9b:43:bd:4d:ec:3e:0e:05:12:66:3a:3d:2e:c5:3f:0a:08:b2:8a:c4

The following clauses describe the content of this XML Schema.

A.4 ASiCManifest content

ASiCManifest content is conformant to the ASiC XML Schema. Here follows an extract of this schema, relevant to ASiCManifest:

```
   <xsd:element name="ASiCManifest" type="ASiCManifestType">
      <xsd:annotation>
         <xsd:documentation>Schema for ASiC-E with CAdES and/or Time-Stamp tokens, specifying
content for ASiCManifest.xml</xsd:documentation>
      </xsd:annotation>
   </xsd:element>
   <xsd:complexType name="ASiCManifestType">
      <xsd:sequence>
         <xsd:element ref="SigReference"/>
         <xsd:element ref="DataObjectReference" maxOccurs="unbounded"/>
         <xsd:element name="ASiCManifestExtensions" type="ExtensionsListType" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:element name="SigReference" type="SigReferenceType"/>
   <xsd:complexType name="SigReferenceType">
      <xsd:attribute name="URI" type="xsd:anyURI" use="required"/>
      <xsd:attribute name="MimeType" type="xsd:string" use="optional"/>
   </xsd:complexType>
   <xsd:element name="DataObjectReference" type="DataObjectReferenceType"/>
   <xsd:complexType name="DataObjectReferenceType">
      <xsd:sequence>
```

```
      <xsd:element ref="ds:DigestMethod"/>
      <xsd:element ref="ds:DigestValue"/>
      <xsd:element name="DataObjectReferenceExtensions" type="ExtensionsListType" minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="URI" type="xsd:anyURI" use="required" />
   <xsd:attribute name="MimeType" type="xsd:string" use="optional" />
   <xsd:attribute name="Rootfile" type="xsd:boolean" use="optional" />
</xsd:complexType>
<xsd:complexType name="AnyType" mixed="true">
   <xsd:sequence minOccurs="0" maxOccurs="unbounded">
      <xsd:any processContents="lax"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name="Extension" type="ExtensionType"/>
<xsd:complexType name="ExtensionType">
   <xsd:complexContent>
      <xsd:extension base="AnyType">
         <xsd:attribute name="Critical" type="xsd:boolean" use="required"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ExtensionsListType">
   <xsd:sequence>
      <xsd:element ref="Extension" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

Here follows the description of all the xml tags defined in this schema:

- `ASiCManifest`: root element. It defines, with all the elements it includes, the content of ASiCManifest.xml metadata used in ASiC-E with CAdES and/or Time-Stamp tokens. It contains one element `SigReference` and one or more `DataObjectReference`. Additional `Extension` elements can be added inside in the optional `ASiCManifestExtensions` element to extend the semantic at the root schema level.

- `SigReference`: this element contains an `URI` element pointing to the CAdES signature or the time-stamp token that applies to the ASiCManifest.xml metadata and the related `MimeType`.

- `DataObjectReference`: this element contains an `URI`, a `MimeType` and an optional Rootfile, referencing respectively a data object, its MIME type a Boolean Rootfile attribute that, if present and set to "true" specify it is a root file with same meaning specified by OCF clause 3.5.1; this element contains `ds:DigestMethod` and `ds:DigestValue` set with the digest algorithm and the related HASH value calculated on the data object. There is a `DataObjectReference` element for each data object referenced by ASiCManifest. Other `Extension` elements can be added in the optional `DataObjectReferenceExtensions` element to extend the semantic associated to each data object referenced by this schema.

- `Extension`: this element can contain an arbitrary content, provided it is well formed XML, that can be used to extend the semantic of this schema.

The `ds:DigestMethod` and `ds:DigestValue` elements are defined in XMLDig.

A.5 XAdESSignatures content

XAdESSignatures is an XML schema to include any number of ds:Signature elements in a single valid XML data and can be also used outside the context of the present document for the same purpose.

XAdESSignatures content is conformant to the ASiC XML schema attached to the present document as specified in clause A.3 and here follows an extract, relevant to XAdESSignature:

```
<xsd:element name="XAdESSignatures" type="XAdESSignaturesType">
   <xsd:annotation>
      <xsd:documentation>Schema for parallel detached XAdES Signatures </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:complexType name="XAdESSignaturesType">
   <xsd:sequence>
      <xsd:element ref="ds:Signature" maxOccurs="unbounded"/>
```

```
    </xsd:sequence>
</xsd:complexType>
```

The root element `XAdESSignatures` contains one or more `ds:Signature` elements containing each a detached signature conformant to XAdES.

A.6 Naming and referencing data within ASiC

ZIP format specification [5] defines a cross-platform file storage and transfer format. In the present document the terms "data object" or "metadata" indicate information stored in the container using the ZIP format without assuming any particular data storage technology.

Data objects and metadata can be hierarchically structured in folders as allowed by files in the ZIP format.

Valid data object and metadata naming shall comply with the ZIP specification [5] and any applicable specific container specification the implementation claims to be compliant with. In the present document the character "*" used to indicate an arbitrary character string of any length, including zero, chosen by the specific implementation to compose a valid file name.

EXAMPLE: "xxxx/yyyy*.ext" indicates a data object in the "xxxx" folder whose name begins with "yyyy" followed by any allowed character and terminating with ".ext". Possible values that comply with this example are "xxxx/yyyy.ext" or "xxxx/yyyy-1234.ext".

In ASiC containers signed data objects are referenced using ASiCManifest (defined in clause A.4) for ASiC-E with CAdES and according to clauses 5.2.2 item 3c for ASiC-S using XAdES and clause 6.2.4 for ASiC-E with XAdES.

The following rules shall apply to these references, expressed as URIs (as defined in [7]):

1) Reference to data objects within the container shall be relative URIs and the rules specified in ODF [6] clause 3.7 shall apply;

2) Relative URIs present in metadata stored in the "META-INF" folder containing a relative path shall be resolved considering the root directory as the base URI, not taking into account the "META-INF" folder where signature metadata are stored.

Reference to data objects outside the container shall not be allowed.

ASiC signatures are located in META-INF folder. According to these rules, references to data objects internal to the container are relative to the root folder: for example, to reference a data object named "document.xml" in the root directory, correct references are "document.xml" or "/document.xml".

When an ASiC-E container implementation claims compliance to an external technical specification or standard (according to clause 7.2.3) that define different referencing rules, they shall prevail.

NOTE: For referencing data objects for different purposes than electronically signing them, implementers are encouraged to use the same rules defined in this clause (considering that rule in item 2 can be extended to any metadata contained in META-INF) when applications do not have specific requirements to implement different rules. These rules, in fact, are compatible with the rules defined in ODF [6].

Annex B (informative): Example Application to Specific File Formats

Examples of the application of the present document to use case are described in this annex.

This includes:

- ASiC-S: use cases to associate to a single data object, that itself can be a container, one or more detached signatures or a time-stamp token that applies to it and to combine them in a container
- ASiC-E with XAdES: use cases to have one or more XAdES signatures each applicable to a set of data objects. Additional metadata, applicable to data object, can also be protected by signature
- ASiC-E with CAdES: use cases, same as the use cases above, but applying CAdES signatures or time stamping to a set of data objects.

B.1 Examples of ASiC-S

A very common requirement is to hold together a single document with its detached digital signature or its document time-stamp. This example is for a PDF document with a CAdES signature. The equivalent structure may be applied to any form of single document or data format including XML, spreadsheet, TIFF, graphic, video. Similarly, the ASiC-S structure can use other form of signature or time-stamp token including XAdES and RFC 3161 [3] time-stamp token.

ASiC-S offers a standardized solution suitable for any use case where one or more signatures, or time-stamp tokens are applied to a single data object.

B.1.1 PDF document Associated with CAdES Signature

This use case proposes a way to associate a PDF document with an advanced signature that applies to it.

The following data objects are put in the container:

- "mimetype", containing "application/vnd.etsi.asic-s+zip"
- "file1.pdf", containing the PDF document to be signed
- "/META-INF/signatures.p7s" containing the CAdES signature created from the hash value of file1.pdf.

B.1.2 Simple document time stamp

If there is the need to associate a time-stamp token with a data object to which the time-stamp token applies, the following data objects are put in the container:

- "mimetype", containing "application/vnd.etsi.asic-s+zip"
- "file1.pdf", containing the PDF document time-stamped
- "/META-INF/timestamp.tst" containing the RFC 3161 [3] time-stamp created from the hash value of file1.pdf.

B.1.3 Signature of a ZIP file

A possible variant of the example given above is where several documents, in the same or different formats, are all to be signed with the same signature or time-stamp token. In this example, a CAdES signature is used. The same basic structure can be used with a XAdES signature or time-stamp token.

A set of documents to be signed are placed together in a ZIP file named "inner-container.zip". For example:

- "file1.pdf", containing a PDF document

- "file2.xml", containing XML data
- "picture3.png" containing a graphical object.

This is placed in an outer container "outer-container.zip" as follows:

- "mimetype", containing "application/vnd.etsi.asic-s+zip"
- "inner-container.zip", with the container to be signed
- "/META-INF/signatures.p7s" containing the CAdES signature created from the hash value of inner-container.pdf.

B.2 Example of ASiC-E with XAdES

In this example more than one document or data object has to be signed with a XAdES signature.

In this example, two XML documents are signed and placed in a container.

The container is produced in a ZIP that includes:

"mimetype", containing "application/vnd.etsi.asic-e+zip".

"file1.xml", containing the first XML document.

"file2.xml", containing second XML document.

"/META-INF/signatures.xml" containing two signatures, the first signing file1.xml and file2.xml and the second signing only file1.xml, as described below.

META-INF/signatures.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<asic: XAdESSignatures
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:asic="http://uri.etsi.org/02918/v1.2.1#"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xades="http://uri.etsi.org/01903/v1.3.2#">
    <ds:Signature>
        <!-- ... -->
        <ds:Reference URI="file1.xml">
            <!-- ... -->
        </ds:Reference>
        <!-- ... -->
        <ds:Reference URI="file2.xml">
            <!-- ... -->
        </ds:Reference>
        <!-- ... -->
    </ds:Signature>
    <ds:Signature>
        <!-- ... -->
        <ds:Reference URI="file1.xml">
            <!-- ... -->
        </ds:Reference>
        <!-- ... -->
        <!-- ... -->
    </ds:Signature>
</asic:XAdESSignatures>
```

B.3 Example of ASiC-E with CAdES

In this example more than one document or other type of data object is to be signed with a CAdES signature and time-stamped. In this case, a different set of documents are to be protected by the time-stamp token from those that are signed. Many examples exist with a number of signatures/time-stamp tokens protected different data objects, for example when documents progress through a workflow.

In this example, two XML documents are signed and placed in a container. Subsequently, PDF transforms of the two XML documents are produced and to bind them to the XML documents both the XML documents and the PDF documents are time-stamped.

The first version of the container is produced in a ZIP file containing:

"mimetype", containing "application/vnd.etsi.asic-e+zip".

"file1.xml", containing the first XML document.

"file2.xml", containing second XML document.

"/META-INF/ASiCmanifest1.xml" containing the hash of file1.xml and file2.xml.

"/META-INF/signatures1.p7s" containing the CAdES signature of "/META-INF/asicmanifest1.xml.

Subsequently the container is updated and the ZIP file is updated to contain (added content in bold):

"file1.xml", containing the first XML document.

"file2.xml", containing second XML document.

"file1.pdf", containing the first document in PDF.

"file2.pdf", containing the second document in PDF.

"/META-INF/ASiCmanifest1.xml" containing the hash of file1.xml and file2.xml.

"/META-INF/signature.p7s" containing the CAdES signature of "/META-INF/asicmanifest1.xml.

"/META-INF/ASiCmanifest2.xml" containing the hash of file1.xml, file2.xml as well as file1.pdf, file2.pdf.

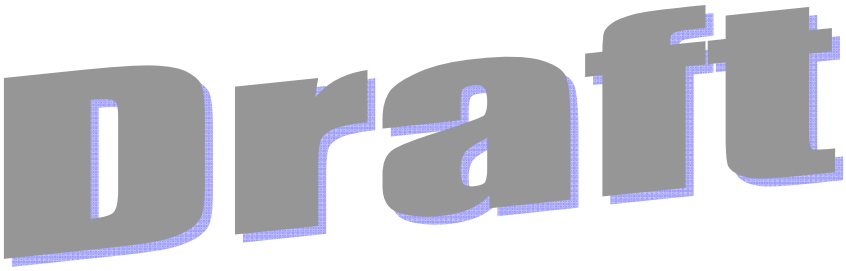
"/META-INF/timestamp.tst" containing the time-stamp token applied to "/META-INF/ASiCmanifest2.xml.

META-INF/ASiCmanifest1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<asic:ASiCManifest
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:asic="http://uri.etsi.org/02918/v1.2.1#"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
   <asic:SigReference URI="META-INF/signature.p7s"
     MimeType="application/x-pkcs7-signature"/>
   <asic:DataObjectReference URI="DOCUMENTS/file1.xml" MimeType="application/xml">
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlenc#sha256"/>
      <ds:DigestValue>j6lwx3SAvKTMUP4NbeZ1</ds:DigestValue>
   </asic:DataObjectReference>
   <asic:DataObjectReference URI="DOCUMENTS/file2.xml" MimeType="application/xml">
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlenc#sha256"/>
      <ds:DigestValue>h3isbr37GE6Ek2wa</ds:DigestValue>
   </asic:DataObjectReference>
</asic:ASiCManifest>
```

META-INF/asicmanifest2.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<asic:ASiCManifest
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:asic="http://uri.etsi.org/02918/v1.2.1#"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <asic:SigReference URI="META-INF/timestamp.tst"
     MimeType="application/vnd.etsi.timestamp-token"/>
   <asic:DataObjectReference URI="DOCUMENTS/file1.xml" MimeType="application/xml">
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlenc#sha256"/>
      <ds:DigestValue>j6lwx3SAvKTMUP4NbeZ1</ds:DigestValue>
   </asic:DataObjectReference>
   <asic:DataObjectReference URI="DOCUMENTS/file2.xml" MimeType="application/xml">
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlenc#sha256"/>
      <ds:DigestValue>h3isbr37GE6Ek2wauT7J</ds:DigestValue>
   </asic:DataObjectReference>
   <asic:DataObjectReference URI="DOCUMENTS/file1.pdf" MimeType="application/pdf">
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlenc#sha256"/>
      <ds:DigestValue>7GE6Ek3SAvKT3isrvEPO</ds:DigestValue>
   </asic:DataObjectReference>
   <asic:DataObjectReference URI="DOCUMENTS/file2.pdf" MimeType="application/pdf">
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmlenc#sha256"/>
      <ds:DigestValue>br37GTMU3SAvKT3sbr3I</ds:DigestValue>
   </asic:DataObjectReference>
</asic:ASiCManifest>
```



Annex C (informative): Container metadata information cross reference

Specification of container formats define a data object structure with metadata about its content, as referenced in clause 4.2.1.

The elements of the OCF, UCF and ODF formats may be combined for use within ASiC as appropriate.

Table C.1 summarizes and compares the different metadata that can be present in each container.

Table C.1: Container content comparison

Container type Metadata in the container	ASiC-E with XAdES	OCF	ODF	UCF
Mimetype	Optional - The present document defines specific media types	Optional. If present the content is: application/epub+zip as defined in OCF [4].	Optional. If present is set to the media type associated to the OpenDocument content carried by the container	Optional
META-INF/manifest.xml	Optional	Optional	Mandatory	Optional
META-INF/metadata.xml	Optional	Optional	Not present	Optional
META-INF/container.xml	Optional	Mandatory	Not present	Optional
META-INF/ *signatures*.xml	At least one signature is present	Optional; a single file is allowed with name "signatures.xml""	Any file in META-INF containing "signatures"	Optional; a single file is allowed with name "signatures.xml"

Annex D (informative): Bibliography

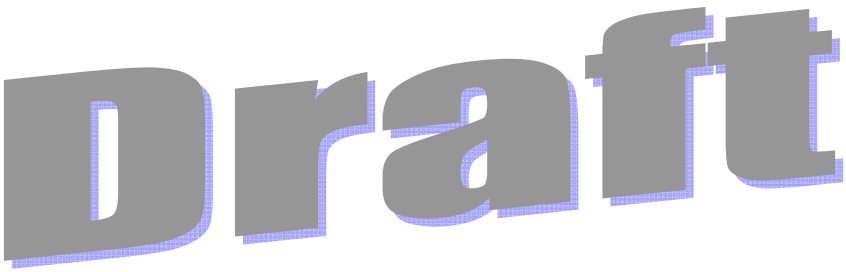
W3C "XSL Transformations (XSLT) Version 2.0".

OASIS "RELAX NG Specification - Committee Specification 3 December 2001".

NOTE: Available at http://www.oasis-open.org/committees/relax-ng/spec-20011203.html.

ETSI Drafting Rules (EDRs).

NOTE: Contained in the ETSI Directives: http://portal.etsi.org/Directives/home.asp.



History

This clause shall be the last one in the document and list the main phases (all additional information will be removed at the publication stage).

Document history		
0.0.0A	July 2013	Initial draft based on TS 101 918 v1.3.1
V0.0.3	November 2013	Stable draft for public review