ETSI TS 102 918 V1.1.1 (2011-04)

Technical Specification

Electronic Signatures and Infrastructures (ESI);
Associated Signature Containers (ASiC)

Reference

DTS/ESI-000084

Keywords

ASiC, e-commerce, electronic signature, security

Contents

Intellectual Property Rights

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Electronic Signatures and Infrastructures (ESI).

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] (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".

TS 101 733 [1] and TS 101 903 [2] define formats for electronic signatures in line with the Directive. 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.

Extending protection for archived document using techniques such as of archive time-stamp following construction of an ASiC container is not addressed by the current version of the present document .

Protection of data objects outside the container is not in scope.

1 Scope

The present document specifies the use of container structures, to bind together a number of signed objects (e.g. documents, XML structured data, spreadsheet, multimedia content) with either advanced electronic signatures or time-stamp tokens into one single digital container. This uses package formats based on ZIP [8] and supports the following signature and time-stamp token formats:

- CAdES (TS 101 733 [1]);
- XAdES (TS 101 903 [2]) detached signature(s);
- RFC 3161 [3] time-stamp tokens.

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

Other time-stamp token formats and methods could be considered in future versions of the present document.

A number of application environments use ZIP based container formats to package sets of files together with meta-information. ASiC technical specification is designed to operate with a range of such ZIP based application environments. Rather than enforcing a single packaging structure ASiC describes how these package formats can be used to associate advanced electronic signatures with any data objects. 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 formats which can be used for basic use cases where a single data object (e.g. a document), or a complete package of data objects have to be signed.
- Clause 6 describes extended formats 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 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;
- the data that contains the certification path and related revocation information to verify the TST.

Reference to documents outside the container and long term verification of time-stamping and CAdES signing of multiple content using the extended form is out of the present document scope.

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 TS 101 733: "Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES)".

[2] ETSI TS 101 903: "Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES)".

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

[5] ISO 32000-1: "Document management - Portable document format - Part 1: PDF 1.7".

NOTE: Available at http://www.adobe.com/devnet/acrobat/pdfs/PDF32000_2008.pdf.

[7] IDPF "OEBPS Container Format (OCF) 1.0".

NOTE: Available at http://www.openebook.org/ocf/ocf1.0/download/ocf10.htm.

[8] PKWARE ".ZIP Application Note".

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

[9] OASIS "Open Document Format for Office Applications (OpenDocument) Version 1.2 Part 3: Packages", (Committee Specification 01).

NOTE: Following publication of final text expected March 2011 the above is to be updated to reference the OASIS standard.

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

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

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

[13] ETSI TS 101 861: "Time Stamping Profile".

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.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in TS 101 733 [1], TS 101 903 [2] and the following apply:

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].

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in ISO 32000-1 [5], TS 101 733 [1] and the following apply:

AdES	Advanced Electronic Signature
ASiC	Associated Signature Container
CAdES	CMS Advanced Electronic Signature
CMS	Cryptographic Message Syntax
OCF	OEBPS Container Format

NOTE: Refer to [7].

ODF	Open Document Format

NOTE: Refer to [9].

OEBPS	Open eBook Publication Structure
TST	Time-Stamp Token

NOTE: Refer to [3].

UCF	Universal Container Format

NOTE: Refer to [i.1].

XAdES	XML Advanced Electronic Signature

NOTE: Refer to [2].

XML	Extensible Markup Language
XMLDSig	XML Signature

NOTE: Refer to [10].

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 applies 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 [8] provides a basic container structure that can associate data objects and their associated signature(s), there is a recognised 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 file objects with related metadata. This includes OCF [7] (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 [9] (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 within a ZIP [8] file. 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 directory structure including:

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

The detached signatures are applied in a way such 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 data 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
- a structure carrying multiple 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 Advances Signatures based on CAdES [1] or XAdES [2] formats or time-stamp tokens conformant to [13].

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

NOTE 2: The present document does not support storage of additional information that might be required for long term verification of time-stamps; this issue can be covered by future versions of the present document.

All ASiC container types support parallel signatures.

4.3 Compliance with external specifications

The extended ASiCtype can be used in a way which is compatible with OCF, UCF and ODF. These containers define a set of files and a structure with metadata about its content included in the "META-INF" folder.

The following are examples of elements from these formats that can be used with ASiC:

- Mimetype file containing the Mime type to identify the selected container type, conforming to relevant clauses in the present document (supported by OCF, UCF and ODF).
- META-INF/container.xml allowing to specify one or more root files to specify how to begin processing the container (mandatory in OCF, not allowed by ODF and not required by UCF).
- META-INF/manifest.xml additional file about container content - (allowed by OCF and UCF and mandatory in ODF).

- META-INF/metadata.xml file that MAY contain user defined metadata associated with data objects (allowed by OCF and UCF not allowed by ODF).
- META-INF/*signatures*.xml file 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 clauses 5 and 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 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 applies to it. This format 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 [8] 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 signature format (i.e. with just 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

- File extension: it is RECOMMENDED to use ".asics" (".scs" is allowed for 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.
- 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.
- The file comment field in the ZIP file header MAY be used to identify the type of a data object within the container. If this field is present, it is RECOMMENDED that this field is set with "mimetype=" followed by the mime type of the data object held in the file.

5.2.2 Contents of the container

This container is built using the following internal structure:

1) An optional "mimetype" inserted containing the mime type defined in clause 5.2.1. If the file extension does not imply use of ASiC then the "mimetype" SHALL be present.

   See clause A.1 for requirements on encoding mimetype.

2) The signed data object in the root level of the ZIP file. It MUST be the only file at the package root level besides the eventual "mimetype" file 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) A file, named "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) A file named "signature.p7s" containing the detached signature of the whole data object specified in item 2 in CAdES format. Multiple parallel signatures and countersignatures are allowed, provided they are all collected in the same CAdES structure; or

   c) A file named "signatures.xml" containing the root element `<asic:XAdESignatures>` as specified in clause A.4, containing one or more detached ds:Signature elements calculated over the whole data object specified in point 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 A.5 SHALL apply. In case referencing attributes (Id, URI or Type) are not present in ds:Reference element then a reference to the signed content is implied. Only standard canonicalization transformation are allowed in ASiC-S.

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.

Other application specific information may be added in further files contained within the META-INFO directory.

If a time-stamp token is applied to an ASiC-S containing a signed package, provided all the verification data is correctly referenced in the time-stamped data object, it can extend the signature validity.

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

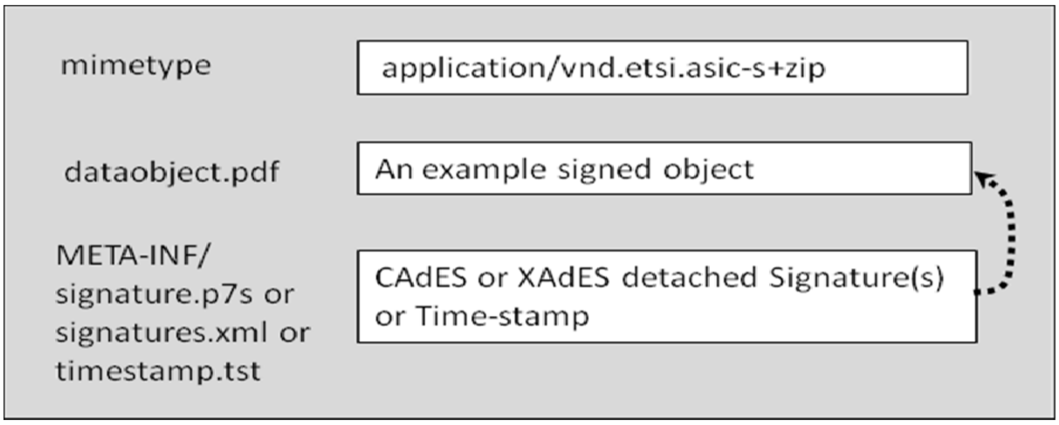

Figure 1: ASiC-S structure applied to a plain file

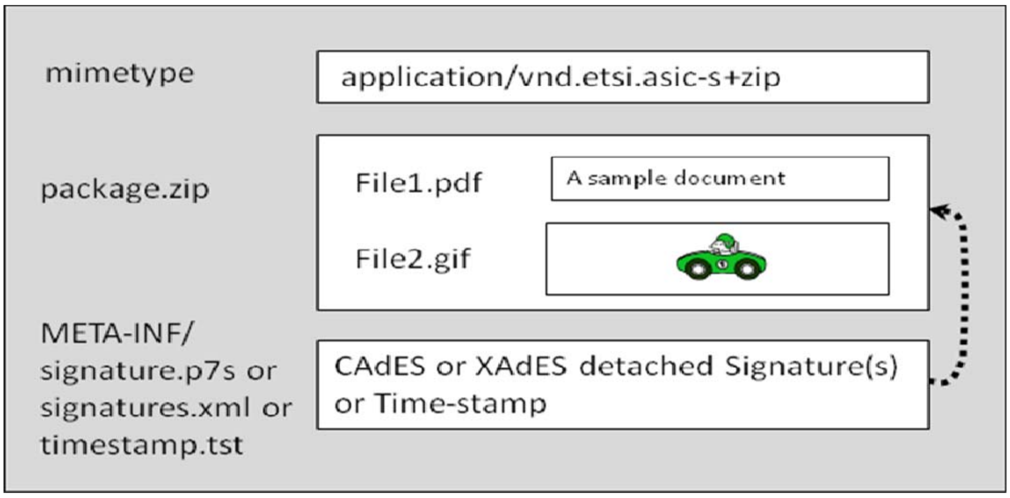

Figure 2: ASiC-S structure applied to a nested container file

5.3 Mime type and file extension correlation check

If the mimetype file is present conformant implementations MUST check that it is consistent with the file extension used for the container.

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 package. The container packages all the mentioned elements. 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.

6.1 General Requirement of ASiC-E

The ZIP format [8] with the structure specified in clause 6.2.2 SHALL be used to bind the contained objects into a single container.

Implementations may support just the ASiC-E signature type for a single signature format (i.e. with just one of CAdES or XAdES); see clause 7.2 for more details.

6.2 Detailed format for ASiC-E with XAdES

One or more signatures files can be present in this format. Each signature file can contain an arbitrary number of ds:Signature elements, each signing an arbitrary set of data objects within the container or external, that SHALL be referenced through URIs, according to rules defined in clauses A.5 and 6.2.4.

The following clauses specify the content of the ASiC-E with XAdES containers.

6.2.1 Media type identification

- File extension: it's RECOMMENDED to use ".asice" (".sce" is allowed for operating systems and/or file systems not allowing more than 3 characters file extensions).
- Mime type: it is RECOMMENDED that the "application/vnd.etsi.asic-e+zip" mime type is used to identify the format of this container.

6.2.2 Contents of Container

Signatures associated to data objects are based on XAdES signatures. See clause 6.2.4 rules regarding use of references to 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 6.1.1. If the file extension does not imply use of a supported container format then the "mimetype" SHALL be present.

2) One or more signature files MUST be present in the META-INF folder containing one or more XAdES signatures conforming to TS 101 903 [2]. The name of each signature file shall contain the term "signatures", with any character placed before or after this term, and have extension ".xml". 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 signature file shall be either:

   a) `<asic:XAdESSignatures>` as specified in clause A4, the RECOMMENDED format; or

   b) `<document-signatures>` as specified in OASIS Open Document Format [9]; or

   c) `<signatures>` as specified in OEBPS Container Format (OCF) [7].

   All the signature files root elements in a given package SHALL be the same.

NOTE: As specified in OCF [7] and ODF [9], in either case the child elements of the root element are one or more <Signature> sibling elements as specified in W3C recommendation: "XML Signature Syntax and Processing" [10].

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

   d) A META-INF/container.xml. If present, the content of this file SHALL be well formed XML conformant to OEBPS Container Format (OCF) [7] specifications. It MUST identify the MIME type and full path of all the root data objects in the container, as specified in OCF (e.g. if an html file that load some external resources such as images is present then the html file is the root data object). Each data object MUST be signed at least by one of the signatures present in the signatures file.

   e) META-INF/manifest.xml. The schema for the XML contained in this file is published at a location specified in OASIS Open Document Format [9].

   f) META-INF/metadata.xml has a user defined content. If present, the content of this file SHALL be well formed XML conformant to OEBPS Container Format (OCF) [7] specifications.

6.2.3 ASiC-E with XAdES example

In figure 3 is represented a typical structure for this container where the XMLDSig 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.

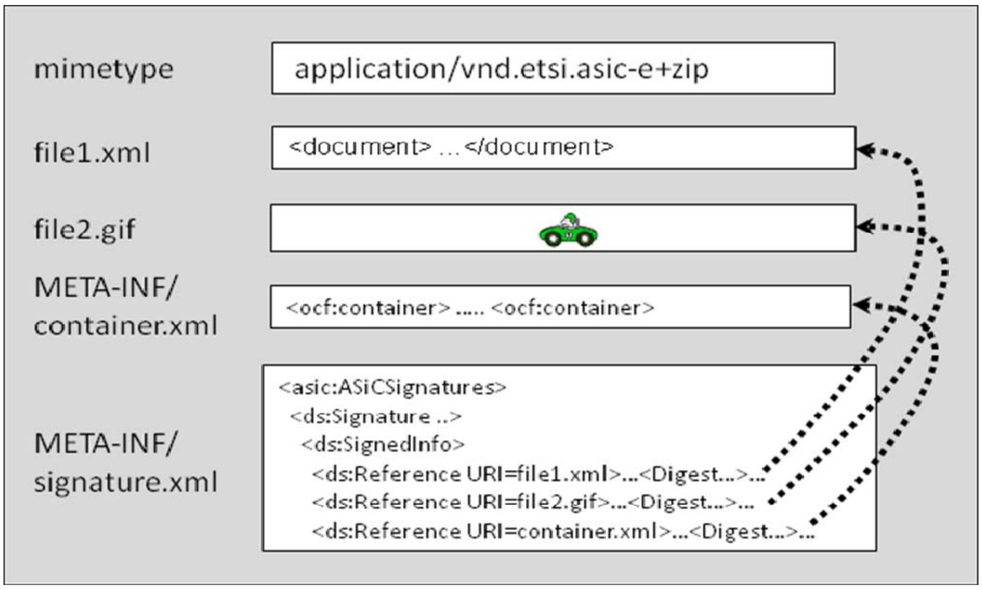
mimetype
application/vnd.etsi.asic-e+zip
file1.xml
<document> ...</document>
file2.gif
META-INF/ container.xml
<ocf:container> ..... <ocf:container>
META-INF/ signature.xml
<asic:ASiCSignatures>
<ds:Signature ..>
<ds:SignedInfo>
<ds:Reference URI=file1.xml>...<Digest...>...
<ds:Reference URI=file2.gif>...<Digest...>...
<ds:Reference URI=container.xml>...<Digest...>...

Figure 3: 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 A.5 SHALL apply.

It is RECOMMENDED that ds:Reference is used to directly reference the signed data objects 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 it is RECOMMENDED to use an Id attribute.

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 can be used to apply CAdES signatures and time-stamp tokens to a set of files.

Each CAdES signature allows presence of parallel signatures and countersignatures.

6.3.1 Media type identification

- File extension: it is RECOMMENDED to use ".asice" (".sce" is allowed for operating systems and/or file systems not allowing more than 3 characters file extensions).
- Mime type "application/vnd.etsi.asic-e+zip".
- The file comment field in the ZIP file header MAY be used to identify the type of a data object within the container. If this field is present, it is RECOMMENDED that is set with "mimetype=" followed by the mime type defined above.

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 6.2.1, see annex C 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, as specified in clause A.3, SHALL be present. For each META-INF/ASiCManifest at least a time-stamp token or a signature that applies to it, in the same META-INF folder SHALL be present.

4) One or more of either:

   a) META-INF/*signature*.p7m containing a CAdES signature conforming to TS 101 733 [1] where *signature*.p7m denotes any string (within the constraints of ZIP or applicable container format) may be placed before and after "signature"; or

   b) META/INF/*timestamp*.p7m containing a binary TimeStampToken as defined in RFC 3161 [3], where *timestamp*.p7m denotes any string (within the constraints of ZIP or applicable container format) may be placed before and after "timestamp".

Implementations claiming conformance to ASiC-E with CAdES, according to clause 7.2.2, or time-stamp token, according to clause 7.3.2, 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 1: Users verifying this container type using a CAdES implementation not aware of ASiC specific rules are warned that any data objects referenced by the signed data may not be checked.

NOTE 2: In order to enable use of existing CMS/CAdES implementations as the basis for ASiC, no specific contentInfo Object Identifier has been defined to identify the specific type of object being signed.

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

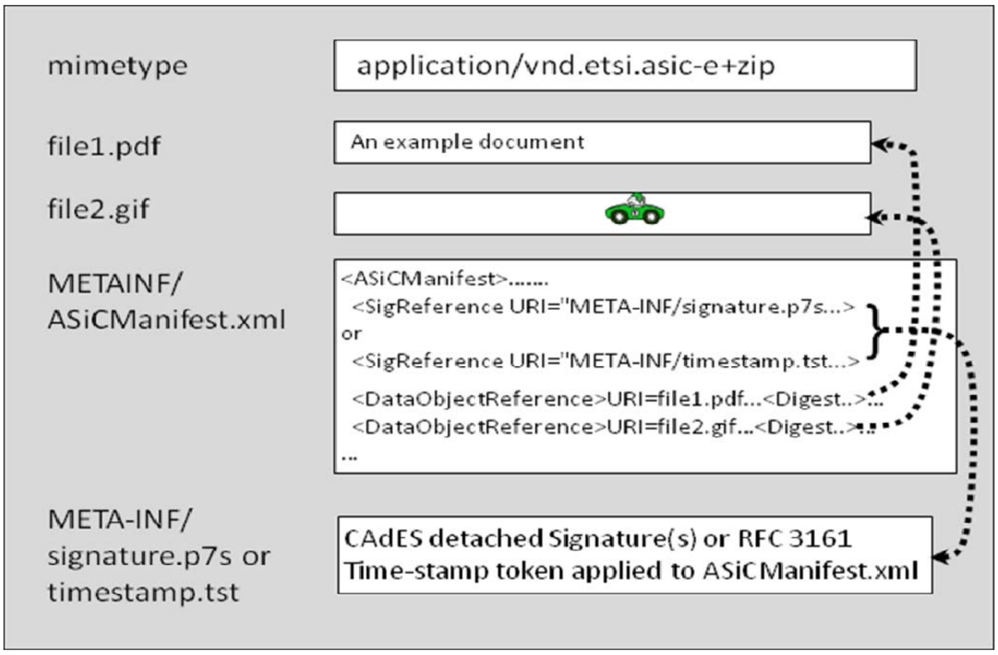
mimetype
application/vnd.etsi.asic-e+zip
file1.pdf
An example document
file2.gif
METAINF/
ASiCManifest.xml
<ASiCManifest>.......
<SigReference URI="META-INF/signature.p7s...>
or
<SigReference URI="META-INF/timestamp.tst...>
<DataObjectReference>URI=file1.pdf...<Digest..>...
<DataObjectReference>URI=file2.gif...<Digest..>...
...
META-INF/
signature.p7s or
timestamp.tst
CAdES detached Signature(s) or RFC 3161
Time-stamp token applied to ASiCManifest.xml

Figure 4: ASiC-E with CAdES signature

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

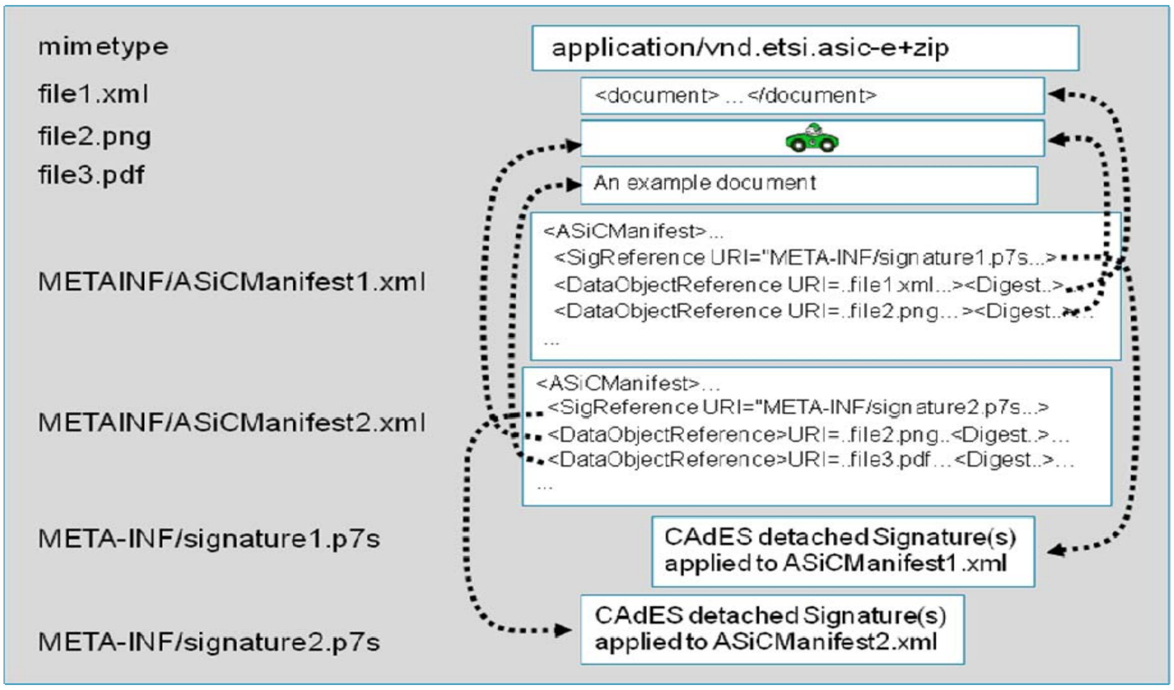
mimetype
application/vnd.etsi.asic-e+zip
file1.xml
<document> ... </document>
file2.png
file3.pdf
An example document
METAINF/ASiCManifest1.xml
<ASiCManifest>...
<SigReference URI="META-INF/signature1.p7s...>
<DataObjectReference URI=..file1.xml...><Digest..>
<DataObjectReference URI=..file2.png...><Digest..>
...
METAINF/ASiCManifest2.xml
<ASiCManifest>...
<SigReference URI="META-INF/signature2.p7s...>
<DataObjectReference>URI=..file2.png..<Digest..>...
<DataObjectReference>URI=..file3.pdf...<Digest..>...
...
META-INF/signature1.p7s
CAdES detached Signature(s) applied to ASiCManifest1.xml
META-INF/signature2.p7s
CAdES detached Signature(s) applied to ASiCManifest2.xml

Figure 5: ASiC-E with CAdES containing different signatures

6.4 Mime type and file extension correlation check

If mimetype file is present conformant implementations MUST check that it is consistent with the file extension used for the container. Also, if the METAINF includes manifest objects containing the mimetype of the object reference (e.g. ASiCManifest) implementations must check that this is coherent with the object referenced.

7 Conformance requirements

The present clause defines conformance requirements for the generation and/or verification of the Associated Advanced Electronic Signatures as specified in the present document.

An implementation can claim conformance to ASiC if it supports at least one of the following clauses. Conformance to any set of the following 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

Implementations claiming conformance to this clause SHALL meet the requirements specified in clauses 5.1 and 5.2, SHALL support the content specified in clause 5.1.2 point 3b and, if a verifying implementation SHALL support clause 5.3.

7.1.2 ASiC-S XAdES conformance

Applications claiming conformance to this clause SHALL meet the requirements specified in clauses 5.1 and 5.2, SHALL support the content specified in clause 5.1.2 point 3c and, if a verifying implementation SHALL support clause 5.3.

7.1.3 ASiC-S Time-stamp token conformance

Implementations claiming conformance to this clause SHALL meet the requirements specified in clauses 5.1 and 5.2, SHALL support the content specified in clause 5.1.2 point 3a and, if a verifying implementation SHALL support clause 5.3.

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

Implementations claiming conformance to this clause SHALL meet the requirements specified in clauses 6.1 and 6.2 and if a verifying implementation SHALL support clause 6.4. Furthermore, the implementation SHALL support:

- At least one of the media type identification specified in 6.2.1 and both are recommended.
- The content specified in clause 6.2.2 point 4a.

7.2.2 ASiC-E CAdES conformance

Implementations claiming conformance to this clause SHALL meet the requirements specified in clause 6.1 and, if a verifying implementation, SHALL support clause 6.4. Furthermore, the implementation SHALL support clause 6.3 with the following exceptions:

- At least one of the media type identification specified in 6.3.1 and both are recommended.
- The content specified in clause 6.3.2 point 4b is excluded, unless conformance to ASiC-E Time-stamp is also claimed.

7.2.3 ASiC-E Time-stamp token conformance

Implementations claiming conformance to this clause SHALL meet the requirements specified in clauses 6.1 and 6.3 (clause 6.3.2 point 4a excluded) and, if a verifying implementation, SHALL support clause 6.4. Furthermore, the implementation SHALL support clause 6.3 with the following exceptions:

- At least one of the media type identification specified in clause 6.3.1 and both are recommended.
- The content specified in clause 6.3.2 point 4a is excluded, unless conformance to ASiC-E CAdES is also claimed.

7.2.4 ASiC-E other container conformance

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 following requirements, that have precedence over the external specification requirements:

- clause 6.1 SHALL be supported;
- clause 6.2.2 point 2 SHALL be supported;
- at least one file containing one or more XAdES signature conforming to TS 101 903 [2] whose name conforms to clause 6.2.2 point 3 SHALL be present.

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

Annex A (normative): ASiC metadata specification and referencing

This annex contains the definition of ASiC specific content or additional specification for metadata already defined in other container formats.

A.1 Mimetype

The "mimetype" file can be used to support operating systems that rely on some content in specific positions in the file (the so called "magic number" as described in RFC 4288 [11] in order to select the specific application that can load and elaborate the file content. The following restrictions apply to the mimetype file for this purpose:

- it has to be the first in the archive;
- it cannot contain "Extra fields" (i.e. extra field length at offset 28 MUST be zero);
- it cannot be compressed (i.e. compression method at offset 8 MUST be zero);
- the first 4 octets 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 position 18.

All multi-octets values are little-endian.

This file MUST NOT be compressed or encrypted.

A.2 MIME registrations

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

NOTE: At the time of publication the MIME-Types are undergoing registration procedure with IANA and users are advised to make their own checks for completion of these formalities (the list of Directories of Content Types and Subtypes 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:	will be none for 8-bit transports and base64 for SMTP or other 7-bit transports.
File extension:	asics or scs.

MIME media type name:	Application.
MIME subtype name:	vnd.etsi.asic-e+zip.
Required parameters:	none.
encoding considerations:	will be none for 8-bit transports and base64 for SMTP or other 7-bit transports.
File extension:	asice or sce.

MIME media type name:	Application.
MIME subtype name:	vnd.etsi.timestamp-token.
Required parameters:	none.
encoding considerations:	will be none for 8-bit transports and base64 for SMTP or other 7-bit transports.
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.
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/v1.1.1#"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns="http://uri.etsi.org/02918/v1.1.1#"
  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 ts_102918v010101p0.zip attached to the present document as a normative part.

The hash values for this file are:

MD-5:	43:1a:a9:d8:ca:12:d1:f4:ab:34:8e:98:58:21:8a:f1
SHA-1:	39:c6:2f:e8:26:a8:1f:d9:57:ab:e5:10:10:9d:55:41:d6:5d:64:5c
SHA-256:	aa:64:2c:60:8d:40:44:15:81:23:0a:0a:d9:b0:a5:72:d8:93:b3:82:d3:2b:10:c4:77:81:f5:b4:83:37:c1:bf

The following clauses describe the content of this XML Schema.

A.4 ASiCManifest content

ASiCManifest is an xml data stored in the META-INF directory of the package in a file whose name MUST start with the text "ASiCManifest" and have ".xml" as extension.

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 file 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 file and the related `MimeType.`

- `DataObjectReference:` this element contains an `URI,` a `MimeType` and an optional Rootfile attributes, referencing respectively a file, its MIME type a Boolean Rootfile that, if present and set to "true" specify it is a root file with same meaning of OCF 3.5.1; this element contains `ds:DigestMethod` and `ds:DigestValue` set with the digest algorithm and the related HASH value calculated on the file. There is a `DataObjectReference` element for each file of the file set 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 ASiCSignatures content

XAdESSignatures is an XML schema to store parallel XAdES Signatures. This can be used outside the context of the present document where detached signatures and related signed data are not in a container.

When used in the context of the present document, XAdESSignatures id stored in the META-INF directory of the package in a file whose name MUST contain the text "signatures" and have ".xml" as extension.

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="ASiCignaturesType">
   <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 Referencing data within ASiC Signatures

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

The following rules SHALL apply to these references, expressed as URIs (as defined in [12]):

1) Reference to data objects within the package SHALL be relative URIs. A specific behaviour is defined for relative URIs in the context of an ASiC container metadata, stored in the "META-INF" folder, as allowed by clause 5.1.4 of [12]:

   a) When the relative URI contains an absolute path, it is resolved relative to the container root directory.

   b) When the relative URI contains a relative path, it is resolved using the root directory as the base URI, not taking into account the "META-INF" folder where signature metadata are stored.

2) Reference to data objects outside the package SHALL be absolute URIs.

ASiC signatures are located in META-INF directory. According to these rules, references to files internal to the container are relative to the root folder: for example, to reference a file 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 that electronically signing them, implementers are encouraged to use the same rules defined in this clause (considering that rule in point 2b can be extended to any metadata contained in the META-INF directory) when applications do not have specific requirements to implement different rules. These rules, in fact, are mostly compatible with the rules defined in OCF, UCF and ODF.

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 signature or a time-stamp token that apply 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 file.

B.1.1 PDF document Associated with CAdES Signature

This use case proposes a secure, informative and unobtrusive manner to package and present a PDF document to a relying party, given that the present document is signed with an advanced signature detached from the PDF document. It is a generic signature platform for the integration of advanced digital signatures, based on the 1999/93/EC [i.3] framework Directive and implementing rules, within an electronic or digitised document: the documents can be PDF or TIFF format.

The following files are packaged together in a Zip file:

- File named: "mimetype", containing "application/vnd.etsi.asic-s+zip".
- File named: "file1.pdf", containing the PDF document to be signed.
- File named: "/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 signed object to which the time-stamp token applies, the package described above is changed to use as time-stamp token as follows:

- File named: "mimetype", containing "application/vnd.etsi.asic-s+zip".
- File named: "file1.pdf", containing the PDF document to be signed.
- File named: "/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 document to be signed are placed together in a ZIP file named "inner-package.zip". For example:

- File named: "file1.pdf", containing the PDF document.
- File named "file2.xml", containing XML data.
- File named "picture3.png" containing a graphical object.

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

- File named: "mimetype", containing "application/vnd.etsi.asic-s+zip".
- File named: "inner-package.zip", containing the packages of documents to be signed.
- File named: "/META-INF/signatures.p7s" containing the CAdES signature created from the hash value of inner-package.pdf.

B.2 Example of ASiC-E with XAdES

In this example more than one document or other type of file is to be signed with a XAdES signature.

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

The package is produced in a ZIP file containing:

- File named: "mimetype", containing "application/vnd.etsi.asic-e+zip".
- File named: "file1.xml", containing the first XML document.
- File named "file2.xml", containing second XML document.
- File named: "/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/v1.1.1#"
  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 file 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 variations of this example could exist with a number of signatures/time-stamp tokens protected different files, for example as 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 package is produced in a ZIP file containing:

- File named: "mimetype", containing "application/vnd.etsi.asic-e+zip".
- File named: "file1.xml", containing the first XML document.
- File named "file2.xml", containing second XML document.
- File named: "/META-INF/asicmanifest1.xml" containing the XML described below containing the hash of file1.xml and file2.xml.
- File named: "/META-INF/signatures1.p7s" containing the CAdES signature created from the hash value of "/META-INF/asicmanifest1.xml.

Subsequently the package is updated and the ZIP file is updated to contain (original content italicised):

- File named: "file1.xml", containing the first XML document.
- File named "file2.xml", containing second XML document.
- File named: "file1.pdf", containing the first document in PDF.
- File named "file2.pdf", containing the second document in PDF.
- File named: "/META-INF/asicmanifest1.xml" containing the XML described below with the hash of file1.xml and file2.xml.
- File named: "/META-INF/signature.p7s" containing the CAdES signature created from the hash value of "/META-INF/asicmanifest1.xml.
- File named: "/META-INF/asicmanifest2.xml" containing the XML described below with the hash of file1.xml, file2.xml as well as file1.pdf, file2.pdf.
- File named: "/META-INF/timestamp.tst" containing the time-stamp token created from the hash value of "/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/v1.1.1#"
  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/xmldsig#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/xmldsig#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/v1.1.1#"
  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/xmldsig#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/xmldsig#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/xmldsig#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/xmldsig#sha256"/>
      <ds:DigestValue>br37GTMU3SAvKT3sbr3I</ds:DigestValue>
   </asic:DataObjectReference>
</asic:ASiCManifest>
```

Annex C (informative): Container metadata information cross reference

Specification of container formats define a set of files and a 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 files that can be present in each container.

Table C.1: Container content comparison

Container type **File in 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 [7].	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 file 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.

History

Document history		
V1.1.1	April 2011	Publication