**CEN/TC XBRL**

Date: 2012-08

**TC XBRL WI XBRL002**

CEN/TC XBRL

Secretariat: NEN

# Improving transparency in financial and business reporting — Metadata container and compliance tools — Part 2 Container

*Einführendes Element — Haupt-Element — Ergänzendes Element*

*Élément introductif — Élément central — Élément complémentaire*

ICS:

Descriptors:

Document type: CWA
Document subtype:
Document stage: Working Document
Document language: E

J:\PROJETS\12-CEN\2012-11-26
Deliverables\Originals\Container\Documentation\WS_XBRL_WI_002_master_copy.doc STD Version 2.4a

# Contents

Page

# Foreword

This document is a working document.

This document is a first draft of the upcoming CWA2 container specification.

# Introduction

CWA2 standardizes two main areas:

1) The way to submit XBRL instances to a regulator, i.e. a container used for the submission of XBRL instances with standardized

   — Encryption;

   — Digital signature;

   — Compression;

2) The way to transmit the supplementary metadata that determine the context of the XBRL instances e.g.

   — the sender of the document;

   — contact details;

   — date and time of submission;

   — …

# 1 Scope

This document specifies a container structure to enable financial institutions to submit their regulatory XBRL reporting to the respective regulators in a standardized way. The container structure allows the packaging and securitization of XBRL data in a uniform way and should lead to a greater transparency and interoperability between the declaring entities and the National and European Supervisory Authorities. The main targeted authorities are the EBA (European Banking Authority) and EIOPA (European Insurance and Occupational Pensions Authority) as well as their related national supervising agencies, but the standard may also be used by other regulators.

The container will be a standardized structure that can contain multiple XBRL instances. It will support compression, encryption and electronic signature; they all will be on the outer container, not on the individual XBRL instance inside the container.

The container definition will not define any transport protocol; submission of a container may thus be freely combined with any type of transport (submission via e-mail, (s)ftp, web portal, web services, …) in accordance with the local requirements of the data taxonomy owner.

The container will not define any file naming conventions, it will only define extensions: XBRL instances will have the .xbrl extension (and not .xml). The signature certificate will be integrated into the container for automatic processing on the authority's side

The deliverables foreseen are:

1) specification of the submission container to an authority and the feedback container from an authority (NORMATIVE);

2) file acknowledge schema for submission containers (NORMATIVE);

   This schema will give all the statuses (positive or negative) indicating if a submission container was well received or if its processing resulted in errors.

   Error conditions will include:

   — electronic signature not valid;

   — certificate used for signature not valid anymore;

   — electronic signature valid, but authentication failed (certificate not accepted for the reporting);

   — decryption failure;

   — decompression failure;

   — local file naming convention not respected;

   — …

3) Reply schema for an XBRL instance in a container (NORMATIVE);

   This schema will give be used to give the processing result of each XBRL instance in the original submission container, including the status of different phases of the validation (positive or negative) as well as potentially a list of error messages indicating why the validation of an instance failed.

   This schema will contain:

— reference to the XBRL instance referred to (filename, receiving datetime, hash, …);

— result of the processing (validation succeeded, validation failed, …);

— list of validation error messages;

— …

4) Compliance tools (NON-NORMATIVE) to be provided will include a free testing environment for the preparers and authorities to ensure full compliance of their containers with the present specification.

## 2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

European Telecommunications Standards Institute (ETSI). Technical Report 102 038 v1.1.1; Electronic Signatures and Infrastructures; XML format for signature policies.

European Telecommunications Standards Institute (ETSI). Technical Specification 101 903 v1.4.1; XML Advanced Electronic Signatures (XAdES).

European Telecommunications Standards Institute (ETSI). Technical Specification 102 176-1 v2.1.1; Electronic Signatures and Infrastructures; Algorithms and Parameters for Secure Electronic Signatures; Part 1 Hash functions and asymmetric algorithms.

CEN Workshop Agreement CWA 14170; Security requirements for signature creation applications.

CEN Workshop Agreement CWA 14167-1: Security requirements for trustworthy systems managing certificates for electronic signatures — Part 1: System Security Requirements

CEN Workshop Agreement CWA 14167-2: security requirements for trustworthy systems managing certificates for electronic signatures — Part 2: cryptographic module for CSP signing operations — Protection Profile (MCSO-PP)

CEN Workshop Agreement CWA 15579: E-invoices and digital signatures

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

Commission Decision 2011/130/EU of 25 February 2011 establishing minimum requirements for the cross-border processing of documents signed electronically by competent authorities under Directive 2006/123/EC of the European Parliament and of the Council on services in the internal market.

Federal Information Processing Standards Publication 186-3: Digital Signature Standard (National Institute of Standards and Technologies, U.S. Department of Commerce).

Federal Information Processing Standards Publication 180-4: Secure Hash Standards (National Institute of Standards and Technologies, U.S. Department of Commerce).

National Institute of Standards and Technologies, Special Publication 800-107, Recommendation for applications using approved hash algorithms.

W3C Recommendation XML Signature Syntax and Processing.

W3C Recommendation XML Encryption Syntax and Processing.

W3C Note XML Encryption Requirements

XBRL International (XII), Extensible Business Reporting Language (XBRL) 2.1, Recommendation – 2003-12-31

PKWARE Inc., APPNOTE.TXT - .ZIP File Format Specification

## 3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

### 3.1

**structural validations**

XML schema validations and structural XBRL validations (XBRL 2.1, Dimensions 1.0 etc. or higher)

### 3.2

**content-related validations**

validations like calculations or formulas

### 3.3

**full container**

a container submitted to an authority containing XBRL instances containing a full set of reporting data set as defined by the authority; it will not only respect all structural validations, but also all content-related XBRL validations

### 3.4

**partial container**

a container submitted to an authority containing XBRL instances covering a subset of the reporting data set defined by the authority; it shall respect all structural validations, but not necessarily all content-related XBRL validations (some of these validations will simply fail because of the lack of some data in the partial container)

### 3.5

**regulator (aka authority)**

a regulatory body that defines the content and structure of filings for the regulated entities as well as the corresponding XBRL taxonomies

### 3.6

**declarer**

role of an entity legally responsible and submitted to regulatory reporting; this role may be assured either with inhouse resources or with the help of technical and functional senders

**3.7**

**technical sender**

role of an entity responsible for creating the header, the package, the compression, the signature, the encryption and the submission (transport) to the Authority

**3.8**

**functional sender**

role of an entity responsible for figures in the reports, i.e. the correct content of XBRL Instance Documents

**3.9**

**authority**

the authority is the receiver of the XBRL Instance documents. The Authority is responsible of receiving of the package, validation and creation of the feedback message to be sent back to the Declarer, with the result of the validations

**3.10**

**initial container**

a container submitted to an authority containing an initial set of XBRL instances for a certain period, entity and reporting type; an initial container will be marked as such in the header XML instance

**3.11**

**update container**

a container submitted to an authority containing XBRL instances covering a correction of the latest initial container sent; it will be marked as such in the header XML instance and is usually a partial container

## 4 Base requirements

XBRL instance documents are created by interested parties (such as banks, investment companies or insurances) mostly because of a legal requirement of declaration. These XBRL instance documents are sent to authorities such as national supervision agencies, national central banks, securities commissions, business registers, etc.

A non empty set of one or more unrepeated XBRL instance documents shall be packed and compressed before submission. Some authorities have reported the need to receive files of about 1GB (uncompressed).

Additionally, a header shall be added to the container to enable the inclusion of additional information and metadata. This header is also being standardized within the present CWA, but it will be treated in a separate CWA document.

This resulting compressed package shall then be signed to ensure the authenticity of the submitter.

The signed and compressed package shall then be ciphered to ensure the confidentiality of the report.

This resulting secured-signed-envelope with the XBRL instance documents and the header is sent to the authority, where it shall be checked in several phases:

— Decryption: From this phase the NSA will obtain the signed envelope;

— Signature: From this phase the NSA will check the authenticity of the message;

— Unpacking and Uncompressing: From this phase the NSA will get the header with the metadata and the XBRL instance documents;

— XBRL validation: a full XBRL validation will be done for each XBRL instance documents.

The authority, after checking the previously defined phases, shall be able to return feedback information about the result of the validation phase, including a hash-code to ensure the non-repudiation of every XBRL instance document validated.

# 5 XBRL instance documents workflow

## 5.1 Major parts and parties in workflow

The parties intervening physically in the reporting process are the declarers, the technical sender and the functional sender and the regulators as defined in chapter 3.

Logically, the communication is limited to an interaction of the declarer with the regulator. There are two ways of that communication:

- submission of a reporting container to the regulator;

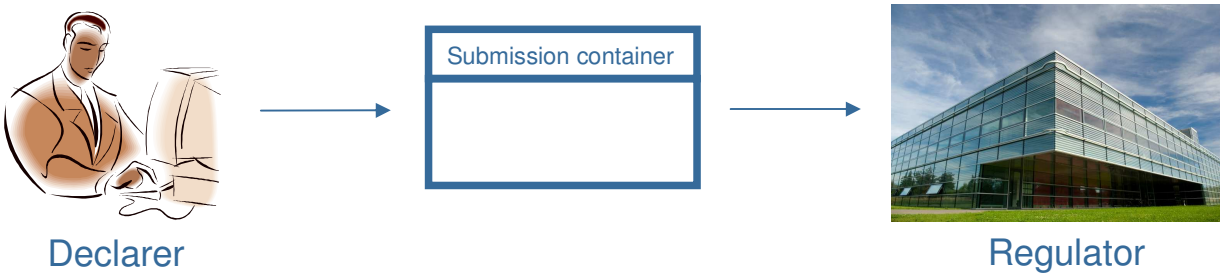


**Figure 1 — Report submission to regulator**

- return of a feedback container to the declarer.

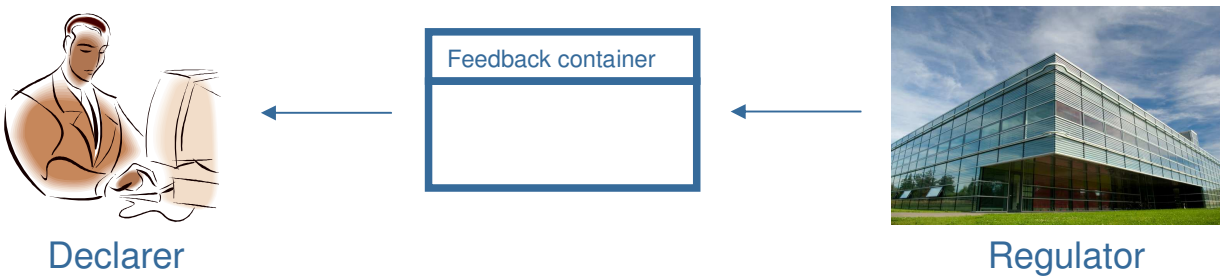

**Figure 2 — Feedback from regulator**

## 5.2 Submission of the report by the declarer

As explained earlier, the main objective is to send in a standardized way a non-empty set of XBRL instance documents. The sender is the declarer, the part required mostly by a legal requirement to generate XBRL instance documents and to send them to the authority.

Figure 3 shows the use cases of issuing XBRL Instance documents to the authority. All these use cases shall be done by the declarer.

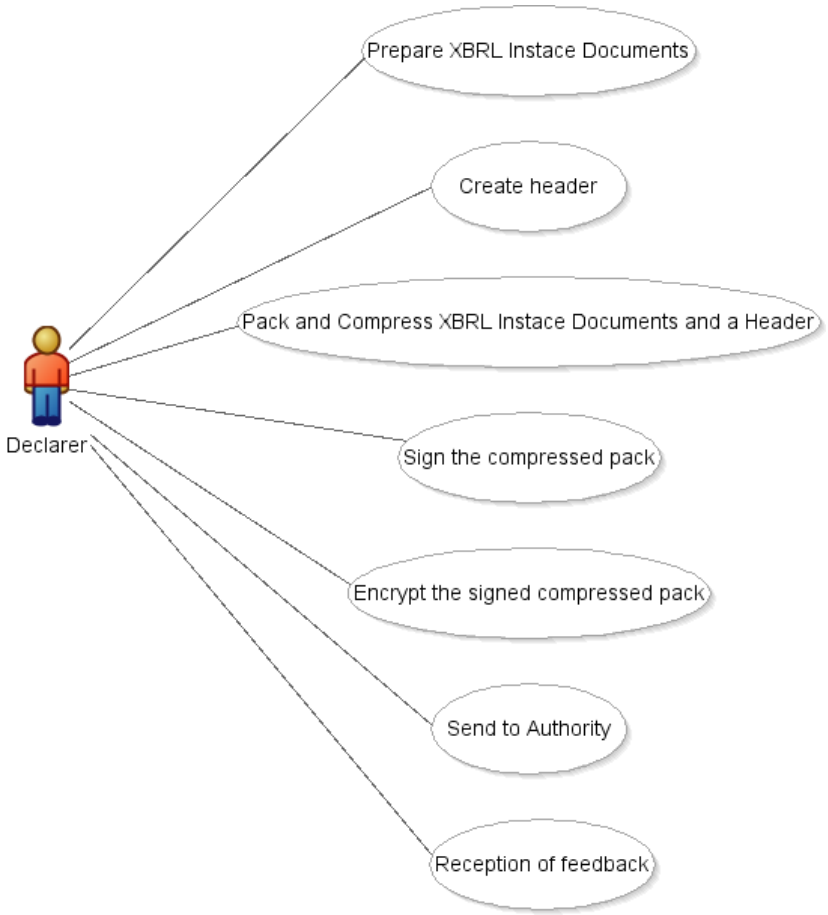


**Figure 3 — Use Cases: Issuing (Declarer)**

### 5.2.1 Use Cases

— Prepare XBRL Instance documents: The declarer shall prepare XBRL instance documents as required by the Authority. These documents shall pass all structural validations and in case of submission of a full container also all content-related XBRL validations. The XBRL Taxonomy shall be provided by the authority;

— Create header: the declarer shall create the header as specified in the second document defined within this CWA;

— Pack and Compress XBRL Instance documents and the Header: the XBRL Instance documents and the header shall be packed and compressed together, according to section "Packaging and compression" of the present document;

— Sign compressed package: To ensure authenticity, the package from the previous use case shall be signed according to European laws. An advanced form of XML signature (XAdES) is used for this purpose. Section "Signature" of the present document specifies this action;

— Encrypt the signed package: Because of the nature of the data sent to the autority, there is sensitive information that shall be protected by an appropiate security level. Chapter "Encryption" of the present document explains the encryption model, based on W3C-XML Encryption;

— Send to authority: The encrypted package shall be sent to the authority;

— Receiving feedback: The authority should create a feedback document with results of validation and a hashcode of every XBRL instance document tested, with the results of the validations.

## 5.3 Receiving of the report by the authority and return of the feedback container

The main objective here is to gain access by the authority of the data inside the XBRL instance documents. The access shall be gained in a secure way, so validations shall be done at different levels to guarantee confidentiality, authenticity, data quality and so forth. Feedback should be generated to inform the declarer about the results of the validations.

Figure 4 presents the use cases for receiving the package. All these use cases shall be done by the authority.


**Figure 4 — Use Cases: Receiving (Authority)**

### 5.3.1 Use Cases

— Receiving: Receive the encrypted container from the declarer;

— Encryption validation: Decrypt the data to obtain the signed package. Note that this is the security layer;

— Electronic signature validation: This validation shall assure the authenticity of the package;

— Uncompress and unpack: By doing this, direct access to XBRL Instance documents can be allowed;

— XBRL Validation: Check that the data in every XBRL Instance Document to be valid against specified Taxonomies. Whether the instance is a part of a full or a partial container, the adequate validation mechanisms shall be used to ensure that the data set represented by the combination of the full container with its potential subsequent partial containers always respects all the structural and content related validations as defined in the taxonomy. In case a specific order in the validation of the instances is required, the order inherent in the listing of the instances inside the header shall be used;

  — Create hash code of validated XBRL instance documents: A hash code of every XBRL instance document will be computed, to allow the declarer to check the integrity of the XBRL instance documents. The results of the XBRL Validation will be together with the hash code of the XBRL instance document being validated;

— Create feedback: An XML feedback file is created for every XBRL instance, reporting the results to the declarer. Section "Feedback" in the chapter "Container format and structure" specifies this part;

  — Create XML feedback file: The feedback should include the results of the validation;

  — Pack and compress feedback.xml: The feedback shall be packed and compressed according to chapter "Packaging and compression" of this document;

  — Sign packed feedback: The package shall be signed by the Authority to assure authenticity. XAdES shall be used as explained in chapter "Signature" of this document;

  — Encrypt signed-packed feedback: In the feedback there can also be sensitive information, so encryption shall be included. In chapter "Encryption" of the present document the encryption model, based on W3C-XML Encryption, is explained;

— Send feedback to the Declarer: The feedback shall be sent to the declarer.

## 5.4 Initial vs. update submissions

An authority shall always allow the submission of initial containers, but need not to allow the submission of update containers.

If an authority only allows the submission of initial containers, all of these containers shall be full containers i.e. they shall contain the full dataset of the reporting as defined by the authority.

If an authority allows the submission of initial and update containers, all update containers should be applied to the last initial container sent. An initial container should mostly be a full container, but the authority may also allow a declarer to start with a partial initial container. Resending an initial container should imply that all initial and update containers sent before should be discarded. An authority allowing the sending of update containers shall provide the necessary mechanisms to ensure that in spite of the possibility of the failure of content-related validations on the update container itself, the combination of the latest initial container with all subsequent update containers should guarantee the full respect of all content-related validations (as defined in the taxonomies) as well as an adequate error handling.

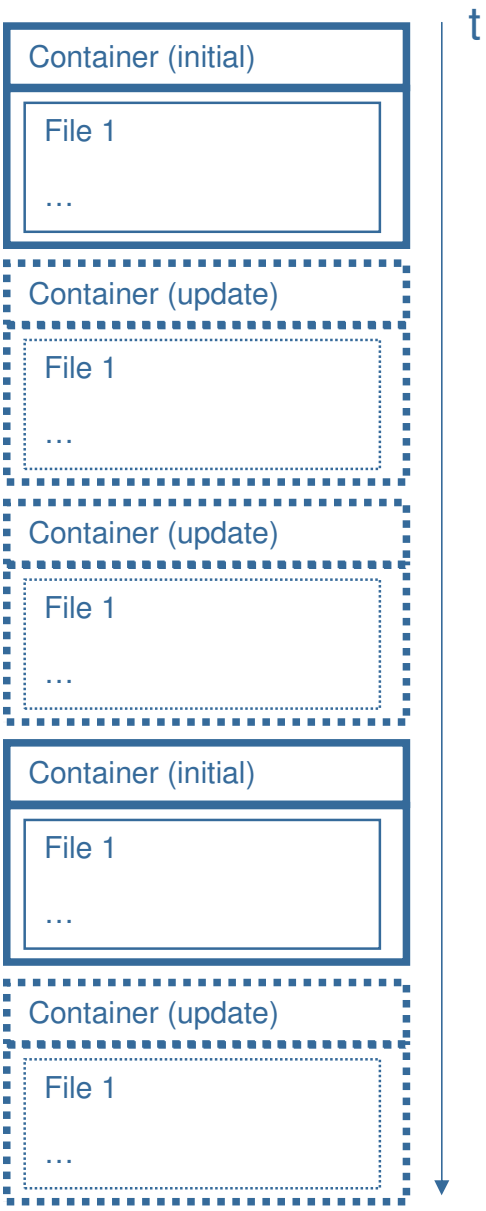


**Figure 5 — Example of a timeline of submission of initial and update containers**

## 6 Container format and structure

The container structure shall use the following layers:

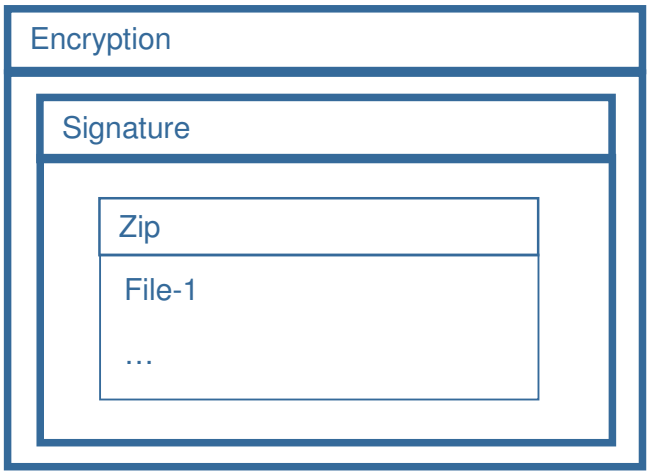


**Figure 6 — Layers and structure of a container**

The standards chosen for the different layers will be described as follows:

— packaging: chapter 7;

— signature: chapter 8;

— encryption: chapter 9.

No standard naming convention applies to the files (neither to the container nor to the XBRL instances in the container); the regulator should fix file naming conventions according to local needs.

Multiple compression packages per security envelope (encryption, signature) are allowed (e.g. for a consolidated reporting for several entities of a group that requires cross-verification)

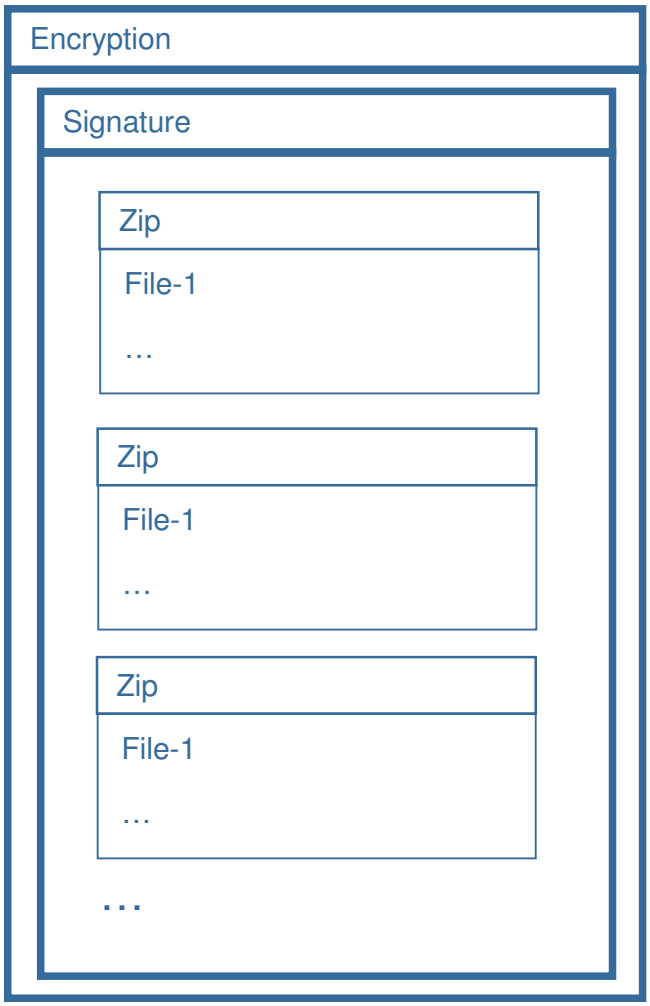


**Figure 7 — Layers and structure of a container containing several data packages**

The multiple XBRL instance documents in the container will share the metadata of the one XML header file; this header is the only file with a naming convention: “header.xml” and it is located on top-level of the compression package. The header lists the XBRL instances contained in the container (in a certain order).

XBRL instances should always have extension .xbrl (neither .xml nor .XML nor .XBRL)

The use of folders is optional within the .zip package; in case they are used, all references (in header to XBRL instances; in XBRL instances to taxonomy files) shall respect them. The folder names used above (“Instances”, “Taxonomy”) are given as examples.

Taxonomy files are optional (they are normally unnecessary and will only be used in case taxonomy extensions by the reporter became allowed in Europe)

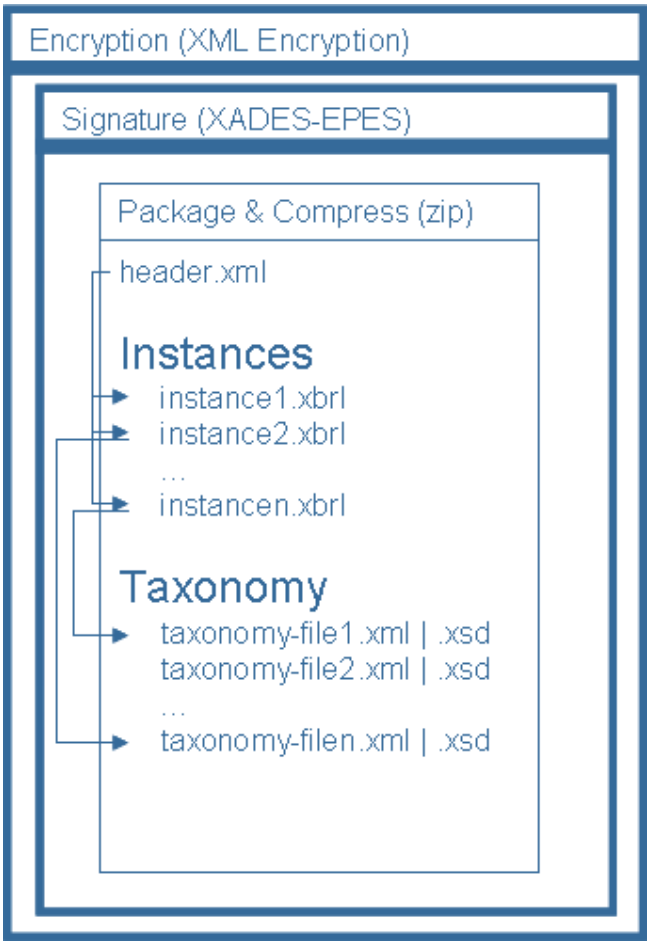


**Figure 8 — Dependencies among files in the container**

### 6.1 Header

Defined in the "Improving transparency in financial and business reporting — Metadata container and compliance tools — Part 1 Header".

### 6.2 XBRL Instance document(s)

### 6.3 Feedback

In case a correct submission container was received, it shall return a feedback container structured as follows:

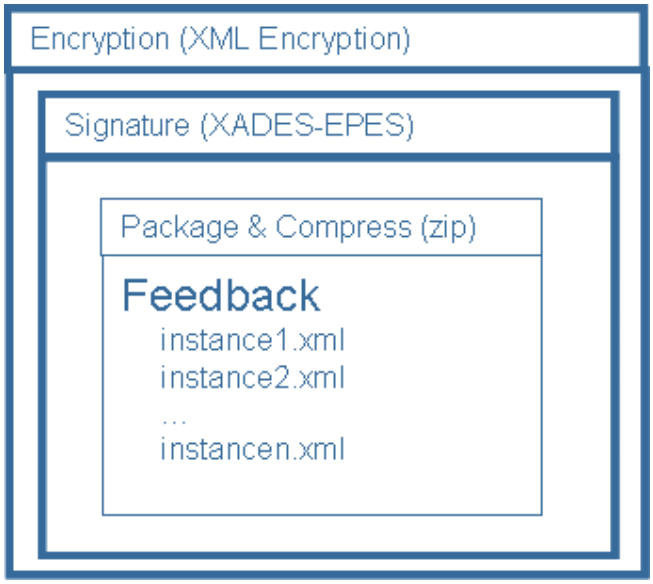


**Figure 9 — Feedback container**

One XML feedback file per XBRL instance in the original submission container will be packaged to the feedback container.

Feedback files will be generated systematically for every submitted XBRL instance, even if no errors at validation time occurred (also positive acknowledge).

The feedback file shall have the same name as the original instance it refers to (but with extension .xml instead of the original .xbrl).

The folder name used here ("Feedback") is given as an example.

### 6.3.1 Scope

Two types of feedback shall be given to a container submitted:

b) Container feedback: a feedback for the entire container (in which either the correct container receiving is acknowledged or in which decompression, decryption, signature verification errors etc. are reported);

c) XBRL Instance feedback: feedback files for each XBRL instance document that was submitted within the original submission container itself.

For both of these validations to be performed, there is an XML Schema definition that should be used when the receiving side validator finds errors and the errors should be communicated back to the declarer of the container. Two XML Schemas are defined here for the validation processes a) and b) respectively. Definitions and technical issues regarding the XML Schemas are given in the following sections 6.3.2 and 6.3.3.

### 6.3.2 Container feedback

This issue will be treated in a future version of the document

#### 6.3.2.1 Namespace

The namespace is http://www.eurofiling.info/eu/fr/esrs/containerFeedback

#### 6.3.2.2 Elements and element types

### 6.3.3 Instance document feedback XML Schema

#### 6.3.3.1 Namespace

The namespace is http://www.eurofiling.info/eu/fr/esrs/instanceFeedback

#### 6.3.3.2 Elements and element types

Elements and types defined in the Instance document feedback XML Schema can be found in the table 7-1.

There is also supplementary information for the elements provided in both tables for:

— Type: the type of the element can be a type defined in the schema or a predefined XML Schema type,

— Occurrence: definition for the occurrence of the given element,

— Usage: a recommendation for the scenario in which the element should be used and

— Description: a narrative explanation for the element.

**Table 1 — Element definitions of the InstanceFeedback XML Schema**

| Element name | Type | Occurrence[a] | Usage | Description |
|---|---|---|---|---|
| InstanceFeedback | | | | the root element, validation consists of separate validation sections |
| InstanceNameReference | text | 1 | | The name of the instance document being validated |
| InstanceHashValue | hash/digest value | ? | | Reference to the calculated hash value of the instance document being validated, it has the "Hash HashAlgorithm " attribute that can be used to indicate which hash algorithm has been used |
| InstanceValidationFlag | boolean | 1 | | true = validation successful<br><br>false = errors found in the validation |
| XMLValidationResult | ValidationResult Type | ? | Only when InstanceValidationFlag is set "false" | The first validation includes, XML, Schema, DTS-validations |
| XBRLValidationResult | ValidationResult Type | ? | Only when InstanceValidationFlag is set "false" and ValidationFlag of XMLValidationResult is set as "true" | The second validation to be executed, XBRL 2.1 Conformance Suite 1.0 validation, taxonomy validation |

| | | | | |
|---|---|---|---|---|
| TransformationResult | ValidationResultType | ? | Only when InstanceValidationFlag is set "false" and ValidationFlag of XMLValidationResult and XBRLValidationResult are set as "true" | Transformation into human readable format, i.e. HTML (inline xbrl), Requires XSLT transformation |
| **ValidationResultType** | sequence | | | |
| ValidationFlag | boolean | 1 | | true = validation successful<br><br>false = errors found in the validation |
| ValidationPhase | text | ? | Only when ValidationFlag is set to "false" | may be used to indicate which phase of the validation was unsuccessful. For instance, validations against Formula linkbase definitions can tumble in validations against accuracy or aspect rules. |
| ValidationErrors | ValidationErrorsType | ? | Only when ValidationFlag is set to "false" | lists all the errors found in the validation |
| **ValidationErrorsType** | sequence | | | |
| ValidationError | ErrorType | 1+ | | The error found |
| **ErrorType** | sequence | | | a generic error type that can be used in all validation sections to define found errors |
| ErrorCode | identification code | 1 | | an error code that can be used to identify the error found |
| ErrorNature | ErrorNatureType | ? | | The nature of the error |
| ErrorLocation | text | 1 | | an expression that can be used to locate the error in the instance document, can be an Xpath sentence or line number |
| ErrorDescription | text | 1 | | a description of the found error |
| ErrorSeverity | ErrorSeverityType | ? | | The severity of the error |
| **ErrorNatureType** | text enumeration | | | Possible values: "Structural" or "Content" |
| **ErrorSeverityType** | text enumeration | | | Possible values: "Info", "Warning", "Error" and "Fatal" |
| [a] Occurence: "1"=occurs exactly once, "?"=occurs once or not at all, "1+"=occurs once or more | | | | |

# 7 Packaging and compression

## 7.1 Packaging and Compression

Users of the CWA Metadata container shall pack and compress the created metadata container by using a method that results in a file format with .zip extension.

The metadata container specification defined in here shall not define any specific software to be used in packaging and compression process.

## 7.2 Limitations and performance

The used .zip file format shall support at least the version 2.0 of the APPNOTE.TXT - .ZIP File Format Specification by PKWARE Inc.[1)] The version 2.0 was selected as the minimum level of conformance as the prior versions do not support folder structures that are used in the CWA Metadata container.

The CWA Metadata container does not impose any other requirements for the compression method to be used. However the report receiving authority should communicate to their reporting entities the highest version of .zip file format specification that is supported by the report receiving system.

# 8 Signature

An electronic signature is data in electronic form which is attached to or logically associated with other electronic subject data and which serves as a method for authentication.

A digital signature is one form of electronic signature that uses a cryptographic transformation of the data to allow the recipient of the data to assure the origin and the integrity of the data, and to provide protection against forgery of the data by the recipient.

Signature is, formally speaking, a value generated by the application of a private key to a message via a cryptographic algorithm such that it has the properties of the signer authentication and message authentication (integrity).

The Major parties involved in the business transaction supported by electronic signature are:

— The signer,

— The verifier,

— Trusted Service Providers (TSP) and

— The arbitrator.

The signer is the entity that creates the electronic signature. When the signer digitally signs over data using the prescribed format, it represents a commitment on behalf of the signing entity to the data being signed.

The verifier is the entity that validates the electronic signature. It may be a single entity or multiple entities.

The Trusted Service Providers are one or more entities that help to build trust relationship between the signer and verifier. They support the signer and verifiers by means of supporting services, including:

— User certificates,

1 ) PKWARE Inc., APPNOTE.TXT - .ZIP File Format Specification, version 6.3.3, section 4.4.3.2. http://www.pkware.com/documents/casestudies/APPNOTE.TXT

— Cross-certificates,

— Time-stamp tokens and

— Certificate Revocation Lists (CRLs), Authority revocation list (ARLs), Online Certificate Status Provider (OSCP) responses.

The Arbitrator is the entity that arbitrates in disputes between a signer and a verifier.

## 8.1 Signature requirements

These are the signature requirements that were taken into account:

— Provide non-repudiation;

— Allows the addition of multiple files to a single signature envelope;

— Compliant with European Directive 1999/93/EC;

— Shall contain the signer's digital X.509 v3 certificate;

— Shall contain the signing time;

— Shall include information about policy to verify electronic signature;

— Avoid the use of MD5 or SHA-1[2)] ;

— Long term validation is not needed, as signature should be validated in a limited time-frame.

## 8.2 Signer

Signature is done using a Public Key Infrastructure (PKI) schema. PKI ensures the following:

— Trusted and efficient management of public and private keys

— Anytime you use a public key, you can be sure that the associated private key is indeed owned by the subject whose public key you are using.

Public-private key combination is at the heart of Public Key Infrastructure, and is based on asymmetric encryption.

The use of public and private keys in digital signature is illustrated in Figure 10. The signer's private key is used to sign the envelope, and the signer's public key is included in the signed envelope in form of a X.509 v3 certificate.

[2] MD5 was broken in August 2004, so it's better no longer use. The security level of SHA-1 has significantly decreased since February 2005, so may be also phased out. Maybe it must be required to use a SHA-2 family algorithm for the digest. (Probably SHA256 or SHA512 for hash functions and RSA, DSA or ECDSA for signature algorithm)

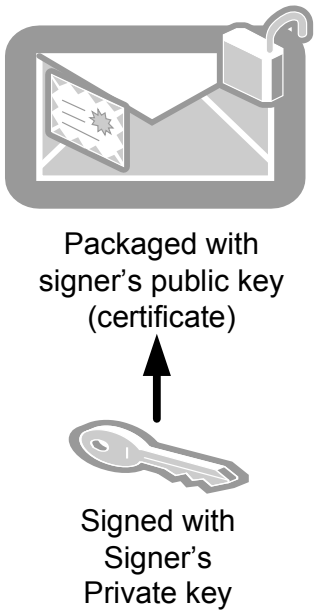


**Figure 10 — Signer's public and private keys in signature**

## 8.3 Signing process

Established requirements involve the usage of an advanced form of W3C XML Digital Signatures (XMLDSIG). The selected form is XAdES.

XAdES complaint with European Directive 199/93/EC, and is one of the three form of Advanced Electronic Signautures (AdES) that comply with the technical specifications set in the Annex of the Commission Decision 2011/130/EU (of 25th February 2011, establishing minimum requirements for the cross-border processing of documents signed electronically by competent authorities under Directive 2006/123/EC of the European Parliament and of the Council on services in the internal market.

In the next sub-chapter specifications about XMLDSIG, XAdES, XAdES-BES and XAdES EPES are included. Note that:

— XAdES is an extension of XMLDSIG;

— XAdES BES is an particular form of XAdES;

— XAdES EPES is an extension to XAdES BES.

The selected standard for digital signing the compressed package is XAdES-EPES.

### 8.3.1 XMLDSIG

XML signatures are digital signatures designed to be used in XML transactions. The standard defines a schema for capturing the result of a digital signature operation applied to arbitrary data. Like non-XML digital signatures (as PKCS), XML signature adds authentication, data integrity and support for non-repudiation to the data that they sign. XML signature has been designed to take advantage of the Internet and XML.

A fundamental feature of XML Signature is the ability to sign only specific portions of the XML tree, rather than the complete document.

A XML signature can sign more than one type of resource. For example, a single XML signature might cover character-encoded data (HTML), binary-encoded data (JPG), XML-encoded data, and a specific section of an XML file.

Signatures are related to data objects via URIs. Within a XML document, signatures are related to local data objects via fragment identifiers.

A signature may be (non-exclusively) described as detached, enveloping or enveloped. **Enveloped or enveloping** signatures over data within the same XML document as the signature. **Detached** signatures are over data external to the signature element.

Enveloped signature is over the XML content that contains the signature as an element. The content provides the root XML document element. Obviously, enveloped signature shall ensure not to include their own value in the calculation of the signatureValue.

When the signature is over content found within an Object element of the signature itself, we are using **enveloping** signature. The object (or its content) is defined via a Reference (a URI fragment identifier or transform).

It's easier to see (in XML digital signature), enveloping signature as signature is parent, and enveloped signature when signature is child. In enveloped signature, a signature element is a descendant of the element being signed.

Figure 11 summarise the structure of a XML signature.

Basic steps to create a XML signature include:

d) Determine which resources are to be signed;

e) Calculate the digest of each resource. Each referenced resource is specified through a <Reference> element;

f) Collect the Reference elements (with their associated digest) within a <SignedInfo> element;

g) Signing: Compute the signature of the <SignedInfo> element, and put the signature value in a <SignatureValue> element;

h) Add key information: If keying information (X509 certificate) is to be included, it will be placed in a <KeyInfo> element. The public key needed to verify signature shall be included here and

i) Enclose in a <Signature> Element. Place the elements <SignedInfo> <SignatureValue> and <KeyInfo> in a element <Signature>.

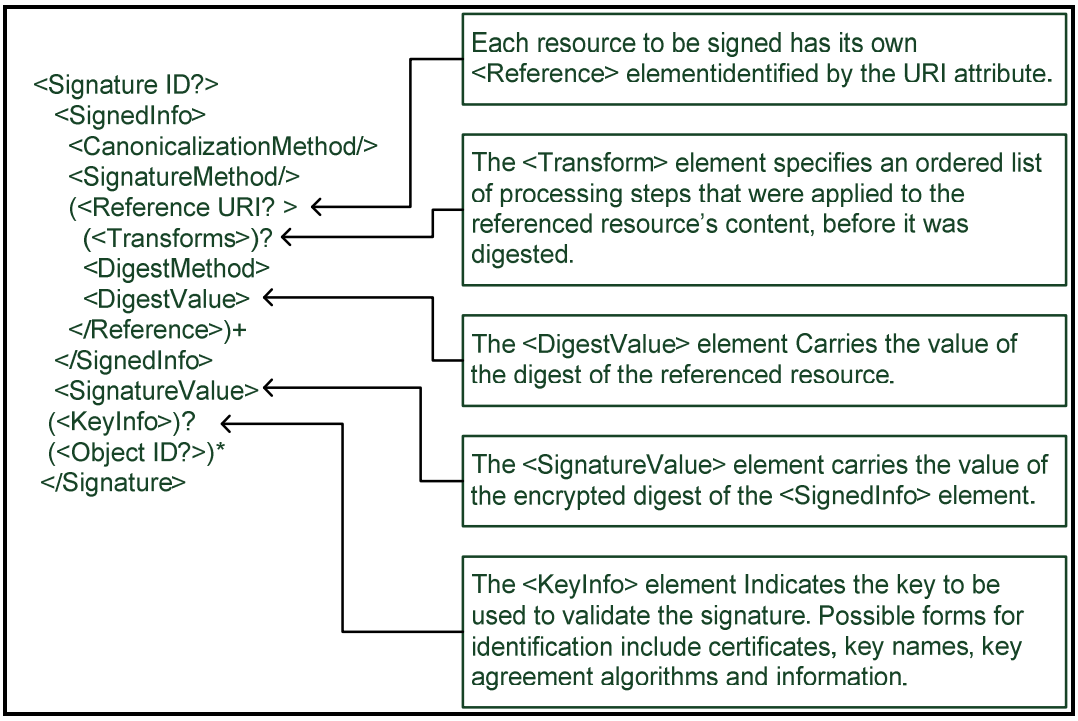


**Figure 11 — Components of an XML Signature**

### 8.3.2 XAdES

XAdES (XML Advnaced Electronic Signature) is built on a XMLDSIG, incorporating qualifying properties defined in ETSI TS 101 903. These properties will be added to XMLDSIG within one ds:Object acting as the bag for the whole set of qualifying properties, or by using the UnsignedProperties element (this element contains a number of properties that not signed by the XMLDSIG signature).

Main characteristics of XAdES are:

— Enables signing of any data, including jpg, pdf, xml, etc.;

— Supports XML package or separate files;

— Support multiple signatures applied in parallel, serial by repeated signing;

— Supports a visual signature appearance (depending on the application);

— Provides long-term validity.

ETSI TS 101 903 defines four forms of XML Advanced Electronic Signatures, namely the Basic Electronic Signature (XAdES-BES), the Explicit Policy based Electronic Signatre (XAdES-EPES), and the Electronic Signature with Validation Data (XAdES-T and XAdES-C).

#### 8.3.2.1 XAdES-BES

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

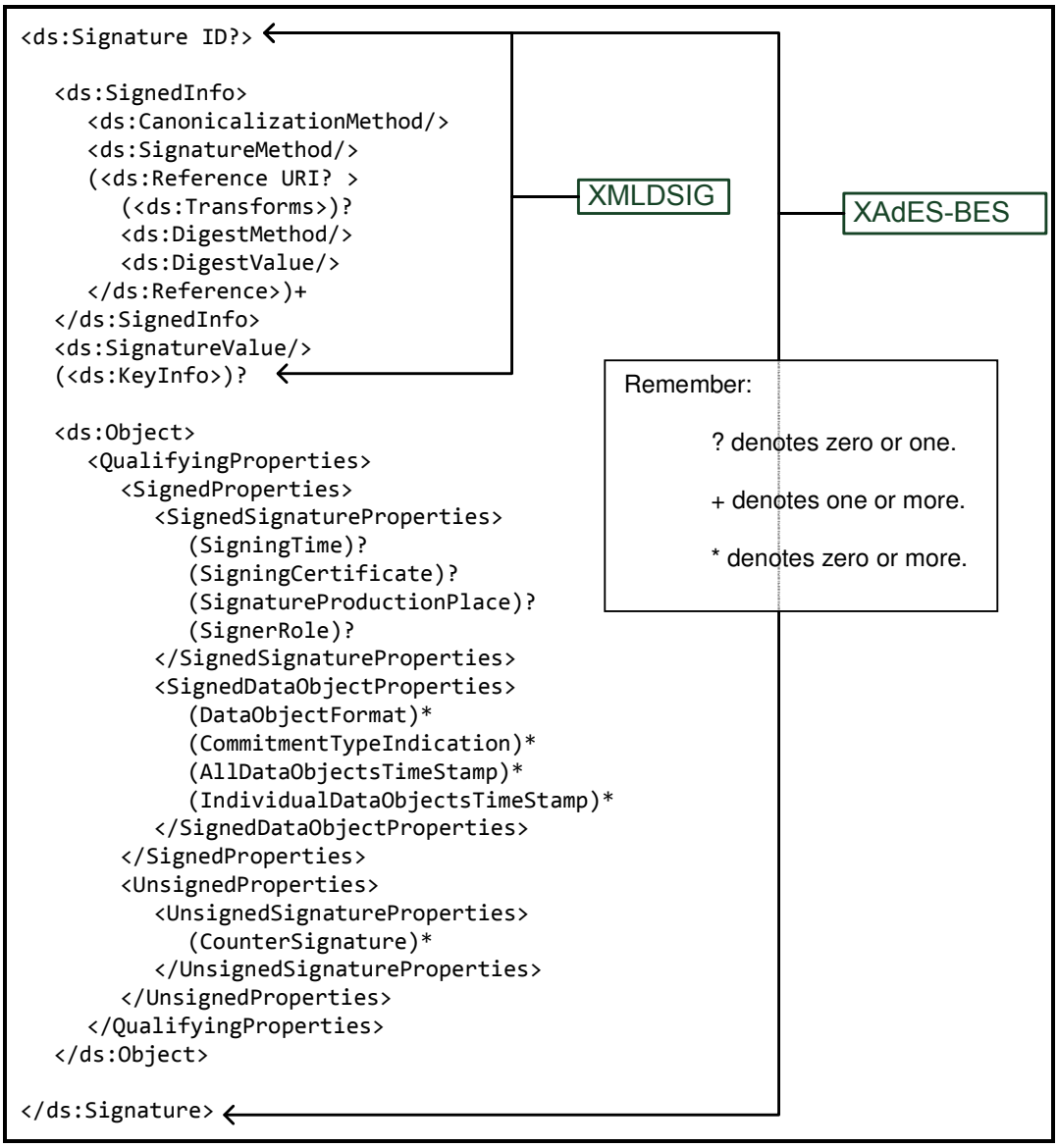


**Figure 12 — Structure of XAdES-BES**

In XAdES-BES, it's mandatory to protect the certificate with the signature, in one (at least) of the following ways:

— Incorporating the SigningCertificate signed property. This property shall contain the reference and the digest value of the signing certificate. It MAY contain references and digest values of other certificates (that MAY form a chain up to the point of trust). In the case of ambiguities identfifying the actual signer's certificate, the applicathions SHOULD include the SigningCertificate property;

— Incorporating the signing certificate within the ds:KeyInfo element, and signing at least the signing certificate.

Commission Decision 2011/130/EU states that the 'SigningCertificate' signed signature property shall contain the digest value (CertDigest) and IssuedSerial of the signer's certificate stored in ds:KeyINFO. The optional URI in 'SigningCertificate' field shall not be used. So, applications may look for the signer certificate in ds:KeyInfo on the basis of hash equivalence.

A XAdES-BES signature may also contain (according ETSI TS 101 903):

- The SigningTime signed property;
- The DataObjectFormat signed property;
- The CommitmentTypeIndication signed property;
- The SignerRole signed property;
- The SignatureProductionPlace signed property;
- One or more IndividualDataObjectsTimeStamp or AllDataObjectTimeStamp signed properties;
- One or more CounterSignature unsigned properties.

Figure 12 shows the structure of XAdES-BES.

XAdES-BES is the minimum format for an electronic signature to be generated by the signer. On its own, it does not provide enough information for it to be verified in the long-term. XAdES-BES provides basic authentication and integrity protection, satisfying the legal requirements for electronic signatures as defined in the European Directive on electronic signatures.

#### 8.3.2.2 XAdES-EPES

XAdES-EPES is the Explicit Policy based Electronic Signature, and extends the definition of an electronic signature to conform to the identified signature policy. XAdES-EPES incorporates the SignaturePolicyIdentifer element (as shown in Figure 6). This signed property indicates that a signature policy shall be used for signature validation. It MAY explicitly identify the signature policy.

Further information on signature policies is provided in ETSI TR 102 038. See above, in CAdES-EPES.

Commission decision 2011/130/EU (Section 1 - XAdES-BES/EPES), specify:

- The signature is conform to with W3C XML Signature specifications;
- The signature shall at least be a XAdES-BES (or EPES) signature as specified in the ETSI TS 101 903 XAdES specifications, v.1.4.1, and complies with all the followin additional specifications;
- The ds:CanonicalizationMehod that specifies the canonicalization algorithm applied to the SignedInfo element prior to performing signature calculations identifies one of the following algorithms only;
    - Canonical XML 1.0 (omits comments):
        - http://www.w3.org/TR/2001/REC-xml-c14n-20010315;
    - Canonical XML 1.1 (omits comments):

- http://www.w3.org/2006/12/xml-c14n11;

— Exclusive XML Canonicalization 1.0 (omits comments):

- http://www.w3.org/2001/10/xml-exc-c14n#.


**Figure 13 — Structure of XAdES-EPES**

— Other algorithms or of 'With comments' versions of the above listed algorithms should not be used for the signature creation but should be supported for residual interoperability for the signature verification;

— MD5 (RFC 1321) shall not be used as a digest algorithm. Signers are referred to applicable national laws, and for the purposes of guidelines to TESI TS 102 176 and to the ECRYPT2 D.SPA.x report for further recommendations on algorithms and parameters eligible for electronic signatures;

- The use of transforms is restricted to the ones listed below:
  - **Canonicalization transforms**: see related specifications above;
  - **Base64 encoding** (http://www.w3.org/2000/09/xmldsig#base64);
  - **Filtering**:
    - *XPath* (http://www.w3.org/TR/1999/REC-xpath-19991116): for compatibility reasons and conformance with XMLDSig;
    - *XPath Filter 2.0* (http://www.w3.org/2002/06/xmldsig-filter2): as a successor for XPath due to performance issues;
  - **Enveloped signature transform:** (http://www.w3.org/2000/09/xmldsig#enveloped-signature);
  - **XSLT** (style sheet) **transform**.
- The ds:KeyInfo element shall include the signer's X.509 v3 digital certificate (its value and not only a reference to it);
- The SigningCertificate signed signature property shall contain the digest value (CertDigest) and IssuerSerial of the signer's certificate stored in ds:KeyInfo and the optonal URI in SigningCertificate field shall not be used;
- The SigningTime signed signature property is present and contains the UTC expressed as xsd:dateTime (http://www.w3.org/TR/xmlschema-2/#dateTime);
- The DataObjectFormat element shall be present and contain MimeType sub-element;
- In the case the signatures used by member states are based on a qualified certificate, the PKI objects (certificate chains, revocation data, time-stamps) that are included in the signatures are verifiable using the Trusted List, in accordance with Commission Decision 2009/767/EC, of the Member State who is supervising or accrediting the CSP having issued the signatory's certificate;
- Figure 14 summarises the specification that a XAdES-BES/EPES signature shall comply with to be supported technically by the receiving member state.

| **XAdES - BES (EPES)** | | **Common Minimum Requirements** |
|---|---|---|
| (ETSI TS 103 903 applies with the following profiled elements) | | |
| *M=Mandatory; O=Optional; R=Recommended; N=Not used* | | |
| ds: Signature ID | M | |
| ds: SignedInfo | M | |
| ds: CanonicalizationMethod | M | *All the following algorithms MUST be supported for signature verification, creation SHOULD restrict to one of these:*<br>*- Exclusive XML canonicalization 1.0: http://www.w3.org/TR/xml-exc-c14n/*<br>*- Canonical XML 1.0: http://www.w3.org/TR/2001/REC-XML-c14n-20010315*<br>*- Canonical XML 1.1: http://www.w3.org/2006/12/xml-c14n11*<br>*Other methods or "#WithComments" versions of the above methods SHOULD NOT be used.* |
| ds: SignatureMethod | M | ***Algorithms:*** *refer to applicable national laws and for guidelines purposes to ETSI TS 102 176 and to ECRYPT2 D.SPA.7 report for further recommendations.* |
| ds: Reference URI | M | One reference to every original data object to be signed (URIs can point to external object as well), + reference to SignedProperties element |
| ds: Transforms | O | *Verifying applications MUST support all following transforms while signature creation application SHOULD restrict the use of those transforms to the following ones:*<br>*- Canonicalization transforms: see above*<br>*- Base64 encoding*<br>*- XPath and XPath Filter 2.0*<br>*- Enveloped signature transform*<br>*- XSLT transforms* |
| ds: DigestMethod | M | ***Algorithms:*** *refer to applicable national laws and for guidelines purposes to ETSI TS 102 176 and to ECRYPT2 D.SPA.7 report for further recommendations.* |
| ds: DigestValue | M | |
| /ds: Reference | | |
| /ds: SignedInfo | | |
| ds: SignatureValue | M | |
| ds: KeyInfo | M | MUST contain X509 certificate (SigningCertificate signed property MUST contain the digest value of this signer's certificate)<br>Signer's certificate certification chain are RECOMMENDED to be provided as a hint for facilitating the validation process (X.509 certificates MUST be provided in this case). |
| ds: Object | | |
| QualifyingProperties | M | |
| SignedProperties | M | M |
| SignedSignatureProperties | M | M |
| SigningTime | M | UTC (xsd: dateTime). |
| SigningCertificate | M | MUST contain the digest value of the signer's certificate stored in ds:KeyInfo and optional URI is omitted (Applications MAY look for/find the signer certificate in ds:KeyInfo on the basis of hash equivalence). |
| SignaturePolicyIdentifier | O | only for EPES form (and for upper forms built from EPES form) |
| Signature ProductionPlace | O | |
| SignerRole | O | |
| /SignedSignatureProperties | | |
| SignedDataObjectProperties | O | |
| DataObjectFormat | M | When this field is used, applications SHALL ensure that data objects are shown to the user accordingly.<br>When used, a MimeType child-element MUST be used. |
| CommitmentTypeIndication | O | |
| AllDataObjectsTimeStamp | O | |
| IndividualDataObjectTimeStamp | O | |
| /SignedDataObjectProperties | | |
| /SignedProperties | | |
| UnsignedProperties | O | |
| UnsignedSignatureProperties | | |
| CounterSignature | O | |
| /UnsignedSignatureProperties | | |
| /UnsignedProperties | | |
| /QualifyingProperties | | |
| /ds: Object | | |
| /ds: Signature | | |
| | | |
| **Signature topology - Packaging signed original files and signatures** | | |
| SignatureEnveloped | | All MUST be supported |
| SignatureEnveloping | | |
| SignatureDetached | | |
| | | |

**Figure 14 — Additional specifications of XAdES-BES (or -EPES) from Commission Decision 2011/130/EU**

## 8.4 Receiver

## 8.5 Validation of signature

Validation of an electronic signature requires:

- A XML advanced electronic signature built on the format defined in W3C "XML-Signature Syntax and Processing" and ETSI "TS 101 903" with the incorporation of additional qualifying information. This XML advanced electronic signature will include:
  - references to the **signed data object(s)** (as specified in W3C "XML-Signature Syntax and Processing" and ETSI "TS 101 903");
  - **signed properties** (provided by the signer);
  - the **signature** itself as defined in W3C "XML-Signature Syntax and Processing" and ETSI "TS 101 903" (see definitions).
- Validation data, which is the additional data needed to validate the electronic signature; this includes:
  - certificates;
  - revocation status information;
  - time-stamp tokens from Time-Stamping Authorities (TSAs).

**Signed data object(s)** is the user's data that is signed. Enveloped signature shall be used, so the signature is over XML content found inside the signed (xml) document.

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

The **Validation Data** may be **collected by the signer and/or the verifier** and shall meet the requirements of the signature policy. Additional data includes CA certificates as well as revocation status information in the form of certificate revocation lists (CRLs) or certificate status information provided by an on-line service.

Additional data also includes time-stamps and other time related data used to provide evidence of the timing of certain events. It is required, as a minimum, that either the signer or verifier obtains a time-stamp over the signer's signature or a record shall be maintained and cannot be undetectable modified, of the electronic signature and the time when the signature was first validated.

The validation process validates an electronic signature. The output status of the validation 8as defined in ETSI TS 101 903 v1.4.1) can be:

- Invalid: Either the signature format is incorrect, or the digital signature value fails verification;
- Incomplete validation: The format and digital signature verifications have not failed, but there is insufficient information to determine if the electronic signature is valid (for example, all the required certificates are not available, or the grace period is not completed);
- Valid: The signature has passed verification and it complies with the signature validation policy.

## 9 Encryption

http://www.w3.org/TR/xmlenc-core/

This issue will be treated in a future version of the document

## 10 Handling of file data container

This issue will be treated in a future version of the document

## 11 Facts and recommendations

This issue will be treated in a future version of the document

# Bibliography

[1] xxx

[2] xxx