CEN/TC XBRL

Date: 2013-06

TC XBRL WI XBRL002

CEN/TC XBRL

Secretariat: NEN

Improving transparency in financial and business reporting — Metadata container

Einführendes Element — Haupt-Element — Ergänzendes Element

Élément introductif — Élément central — Élément complémentaire

ICS:

Descriptors:

Document type: CWA
Document subtype:
Document stage: CEN Enquiry
Document language: E

C:\EBA\CEN\PublicConsultation201307\PublicConsultation\CWA_XBRL_WI002 (E).doc STD Version 2.5a

Contents

Foreword

This document has been prepared by CEN/WS XBRL, the secretariat of which is held by NEN.

This document is currently submitted to a public consultation.

1 Scope

The purpose of this CWA is to propose a standard for submitting data instances to financial regulators in accordance with the section describing this CWA in the Business plan:

— CWA2: "Metadata container" to wrap a submitted XBRL instance document and compliance test;

— Provide a standard Metadata Container to enable XBRL sourcing, with in addition necessary compliance tools to enable all stakeholders to test and ensure full adherence to the technical standards.

Metadata such as sender of the document, contact details, date and time of submission, version, digital signature, etc. are not included in the taxonomies, because they don't belong to the data model. On the other hand, and often for legal reasons, these data are required by national regulators. As a consequence, a variety of national protocols have been engineered, which complicates the processes cross-border institutions, but also prohibit the possibility to create a harmonized European data collection system. Metadata are needed as well for financial reporting as for company legal and economical data. For the digital signature, existing solutions from the business registers, who have a deep expertise of the topic, may be generalized. In order to ensure compliance with the protocol, this project will deliver online tools for all stakeholders to use and to test compliance with the complete set (metadata container and XBRL instance document).

This CWA will provide standard protocols and mechanisms for digital signature, administrative data such as identification of submitter, feedback parameters, versioning of subsequent submissions and encryption, as well as online collaborative tools to ensure compliance.

This document specifies:

— a **submission container** structure to enable financial institutions to submit their regulatory structured reporting to the respective regulators in a standardized way;

— a **metadata** information structure (called « **Header** ») that is part of the submission container structure;

— an adequate negative (or positive) acknowledge to be returned by the regulator to indicate if the submission container was well received by the regulator (or not);

— a **response container** to allow the regulator to return content-treated error messages for the data instances in case errors occurred during any validation phase.

The main targeted authorities are the EBA (European Banking Authority) and EIOPA (European Insurance and Occupational Pensions Authority) as well as their related national supervision agencies, but the standard may also be used by other regulators. All container structures defined allow the packaging and securitization of data in a uniform way, which should lead to a greater transparency and interoperability between the declaring entities and the national and European supervisory authorities.

In the course of the specification process, supplementary requirements were added by stakeholders or authorities concerned, among which:

— The scope of the data instances to be supported has been extended from pure XBRL instances to any type of structured data instances, including XML, CSV, etc.;

— The possibility of a 2-layer (or even multi-layer) submission process: some data instances are to be processed by the receiving authority itself (e.g. a national authority), others may be forwarded to a subsequent authority (e.g. a European one);

— The possibility of using the structures of this CWA in a secure environment i.e. an environment that has its own signature and/or encryption facilities;

— The possibility of adding non-standard metadata if required (extensibility of the metadata header).

An important development approach for this CWA is to be flexible enough to support many different uses in different environments. For this reason some aspects (e.g. types of identifiers for financial institutions) could not be fixed by this standard and they shall be determined for every specific use of this standard via complementary instructions.

The present specification only defines the structures for the container itself, it does not define any transport aspects; the submission of a container may thus be freely combined with any type of transport protocol (submission via e-mail, (s)ftp, web portal, web services, …) in accordance with the local requirements.

2 Terms and definitions

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

3.1
reporting entity
entity submitted to financial reporting and legally responsible for it

Note 1 to entry: In many cases it uses internal resources to play the role of Content Producer and Technical Sender.

Note 2 to entry: The Reporting Entity is also known as 'Declarer', 'Sender', '<ReportingEntity>'.

3.2
technical sender
(potential) sub-contractor in charge of physically sending the data in respect of this CWA (aware of containers, encryption, etc.)

Note 1 to entry: Also known as '<TechnicalSender>'.

3.3
content producer
(potential) sub-contractor in charge of the production of the content of the reporting and responsible for the accuracy of the content

Note 1 to entry: Also known as '<ContentProducer>'.

3.4
security envelope
XML structures surrounding the .zip file(s) after encryption and / or signature phase in accordance with this CWA

3.5
negative acknowledge
information to the sender that his submission container could not be accepted because of error conditions (usually an instance of the « ContainerFeedback » schema with the tag <ContainerValidationFlag> having the value false

3.6
positive acknowledge
information to the sender that his submission container has been accepted for processing of the data instances (usually an instance of the « ContainerFeedback » schema with the tag <ContainerValidationFlag> having the value true

3.7
instructions
supplementary information drafted by the receiver on how exactly to use this CWA for a determined use.

3 Files in containers

3.1 Introduction

The present section describes the files intervening in this standard, starting with simple files and continuing with the composed ones.

3.2 Data files

Data files are files that contain data, whether these data are structured of not. Data files can be any structured files like XBRL or XML instances, but also unstructured files like spread sheets or word processor files. The container controls files described in the next section as well as composed files (files that contain other files) are not part of the data files.

3.3 Container control files

The three types of container control files developed within this CWA are described in the following sections

Caveat: the structure of these files is planned to change up to the final acceptance of this CWA. See section "Creating a submission container".

3.3.1 Header file

A header file is an XML instance of the XML schema located at http://www.eurofiling.info/eu/fr/esrs/header

The function of the header file is describe the characteristics of the data files in the submission.

3.3.2 Container feedback files

A container feedback file is an XML instance of the XML schema located at

http://www.eurofiling.info/eu/fr/esrs/ContainerFeedback

The function of the container feedback file is confirming from the receiver to the sender of the success or not of the submission.

3.3.3 Instance feedback files

Instance feedback files are XML instances of the XML schema located at:

http://www.eurofiling.info/eu/fr/esrs/InstanceFeedback.

The function of the instance feedback file is confirming from the receiver to the sender of the success (or failure) of the validation in the receiver side of some submitted data files, typically XBRL instance documents.

More visual representations of the error conditions of the data file submitted (e.g. documents with links to an external systems representing the errors graphically, spread sheets with "red" cells indicating error locations, …) may be added as Instance Feedback file with the corresponding file extension, either as a complement or as an alternative to the XML Instance Feedback file.

3.4 ZIP compressed file

A Zip compressed file is a set of one or more files compressed together, that is generated in accordance with the standard set in:

http://www.pkware.com/documents/casestudies/APPNOTE.TXT.

3.5 Secured files

3.5.1 Introduction

The following sections describe the files to which security operations have been applied.

3.5.2 Encrypted file

An encrypted file is a file embedded and encrypted in an XML instance of the XML schema referred to:

http://www.w3.org/TR/xmlenc-core/xenc-schema.xsd.

3.5.3 Signed file

An signed file is a file embedded and signed in an XML instance of the XML schema referred to at:

http://uri.etsi.org/01903/v1.4.1/.

3.5.4 External Signature of a file

To be added if required

3.5.5 External Hash of a file

To be added if required

3.6 File naming conventions

3.6.1 Introduction

This CWA has defined the smallest possible file naming conventions as described in the present section. The goal was to give regulators most degrees of freedom to define for their purposes a file name convention that serves best their requirements. So excepted for the reserved names and suffixes described in this section, the receiver's instructions may define adequate file naming conventions for containers, folders, data files etc.

3.6.2 Reserved file names

3.6.2.1 header.xml

The name « header.xml » is exclusively reserved for files of the type « header file »

3.6.2.2 container.zip

The name « container.zip » is exclusively reserved for « submission containers »

3.6.2.3 containerfeedback.xml

The name « containerfeedback.xml » is exclusively reserved for files of the type « containerfeedback.xml »

3.6.3 Instance feedback file name

An instance feedback file name is composed of the original instance file name while changing the last suffix to « .xml » Example: the instance feedback of « datafile.xbrl » will have the file name « datafile.xml »

Other types of "more visual" instance feedback files will change to the appropriate suffix. Example: the "Excel visual" instance feedback of « datafile.xbrl » would have the file name « datafile.xls »

3.6.4 Reserved file name suffixes

All files shall have the « usual » file extension applicable in environments without restriction to the length of the extension: « .xbrl » for XBRL instances, « .xml » for XML instances, « .csv » for comma seperated files etc.

For protection of « header.xml » and « container.zip » reserved file names, the corresponding « .header.xml » and « .container.zip » file name suffixes are also reserved.

3.6.5 Reserved extended suffixes

3.6.5.1 .signed.xml

The file extension « .signed.xml » is exclusively reserved for signed files

3.6.5.2 .encrypted.xml

The file extension « .encrypted.xml » is exclusively reserved for encrypted files

3.6.5.3 .externalhash.xml

To be added if required

3.6.5.4 .externalsignature.xml

To be added if required

4 Container

4.1 General

A container is a ZIP compressed file that contains a set of file/s to be remitted.

A container may also contain other containers. The way of processing containers inside containers is not covered by the present standard

Folders may optionally be used in a container to better structure the files.

Folder conventions are not defined in this document.

4.2 Submission container

A submission container is a container that contains 1 header and 0 or more files and that is to transfer reporting data from a sender to a receiver. Figure 1 shows an example of a simple submission container with only one type of reporting in XBRL format and no use of folders.

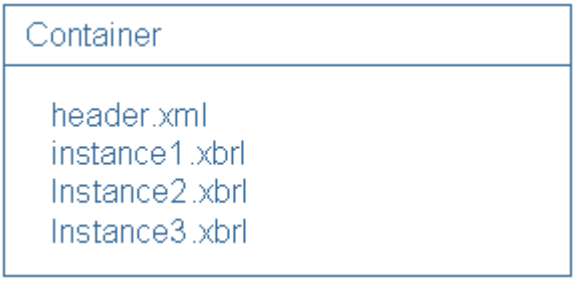

Figure 1 — Structure of a simple submission container

Figure 2 shows an example of an advanced structure of a submission container using folders (bold) to structure multiple types of reporting, containers, supplementary information etc.

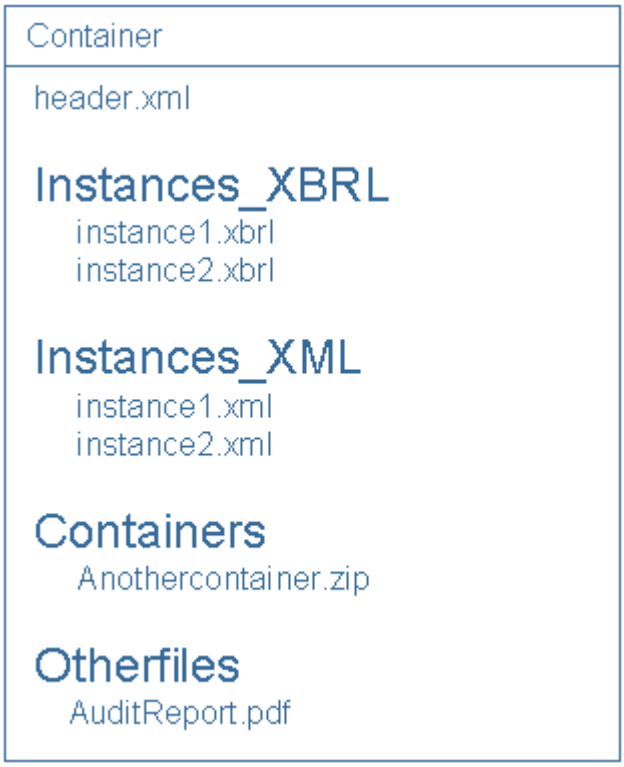

Figure 2 — Structure of an advanced submission container

4.3 Response container

A response container is a container that is to be returned by the receiver of a submission container to its sender to inform the sender about the result of the evaluation of its content (e.g. possible errors).

When applicable (i.e. XBRL instance documents), XML Instances of the InstanceFeedback schema should be used to report the errors that were identified during the validation phase by the receiver, and:

1) One XML instance feedback file should be generated per data instance in the original submission container. Additional “visual” instance feedback files are allowed;

2) XML Instance feedback files should be generated systematically, even if no errors at validation time occurred (so not only negative, but also positive feedback should be provided for any data instance listed in the submission container).

A response container is composed of only one the following file combinations:

— 1 container feedback file;

— 1 to n instance feedback files;

— 1 container feedback file and 1 to n instance feedback files.

Figure 3 shows an example of response container generated on the basis of an incoming submission container with one reporting consisting of 3 XBRL files. All files in the response container are instances of the XML schema InstanceFeedback.

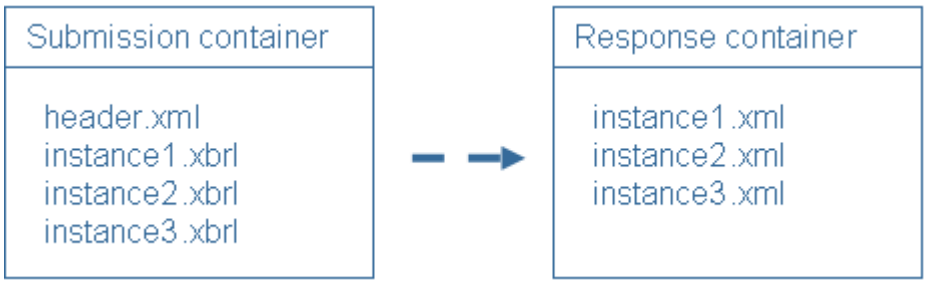

Figure 3 — Response container generated on the basis of an incoming submission container with one reporting

Figure 4 shows an example of a response container generated on the basis of an incoming submission container with 2 different reporting and using folders. All XML files in the response container are instances of the XML schema InstanceFeedback. As a supplement, Excel-type error-diagnostics are returned for Report1

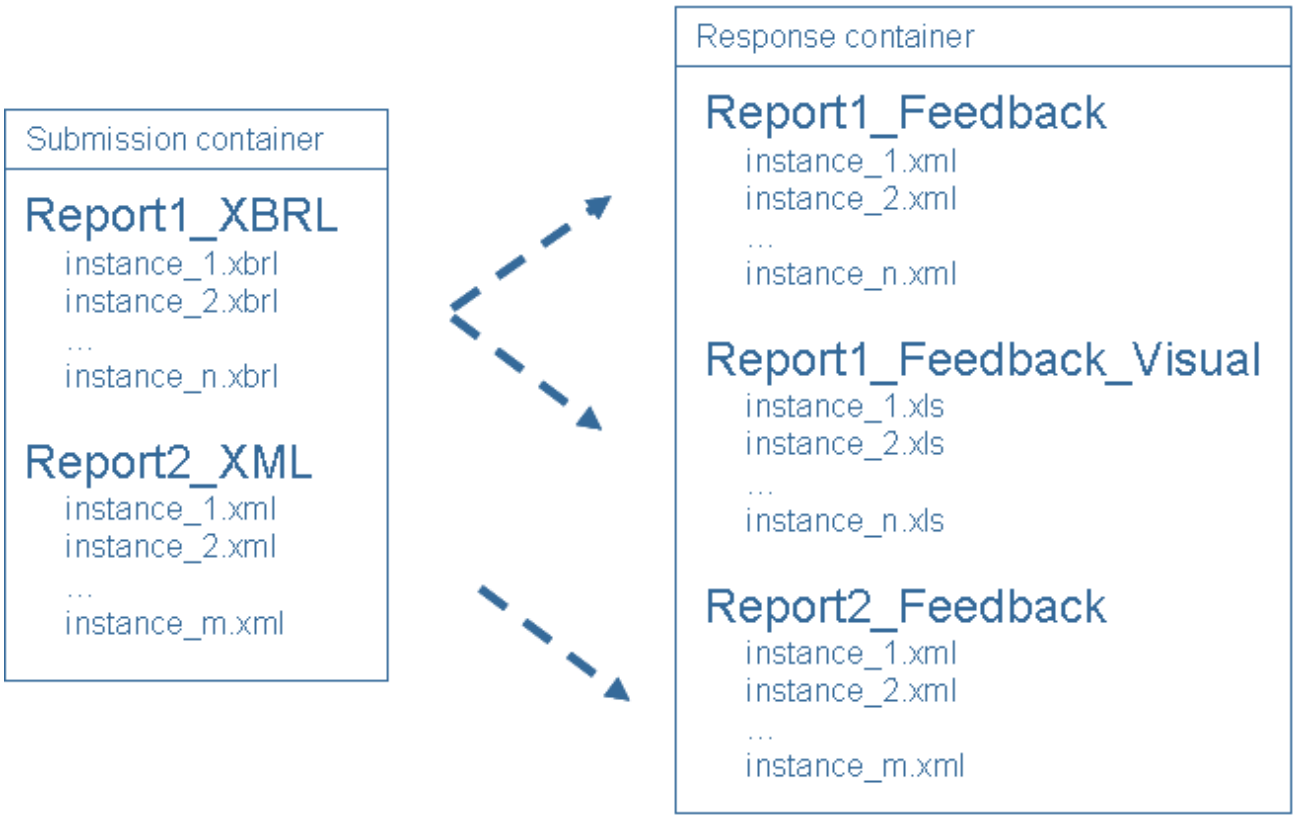

Figure 4 — Response container generated on the basis of an incoming submission container with 2 different reporting and using folders.

5 Primitive functions

5.1 Introduction

The present section describes the primitive functions required to put in place the this CWA.

5.2 Compression functions

Compression is made in accordance with the standard set in:

http://www.pkware.com/documents/casestudies/APPnotE.TXT.

The minimum feature version is 2.0 as defined in Section 4.4.3.2 of the present version of that file (6.3.3).

5.2.1 Creating a ZIP compressed file

Many tools in the market can create ZIP compressed files; interoperability problems are not known as long as multi-volume zip is used. This is why multi-volume ZIP compressed files are not supported by this CWA version.

In order to avoid problems with senders using features of very recent versions not yet supported by the receiver, the instructions of the receiver may fix further constraints on the compression to use (e.g.a maximum level of the zip standard, as supported by the receiver).

5.2.2 Expanding ZIP compressed file

Per analogy previous paragraph.

5.3 Security functions

5.3.1 Introduction

This section describes the primitive functions for signing or encrypting files as well as the way to calculate the hash required in schemata InstanceFeedback and ContainerFeedback.

5.3.2 Encrypting a file

W3C Encryption

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

using key transport RSA-OAEP

http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p

and encrypting data with AES256.

http://www.w3.org/2001/04/xmlenc#aes256-cbc

The selected encryption uses the W3C XML Encryption to cipher a file, embedding it completely into the XML document that will result of the encryption process (there shall be no references to te file at an external location). Inside the CipherData element, there shall be a CipherValue element, but there shall not be a CipherReference element.

Basic steps for encryption are:

— create XML document with the embedded file, using W3C Encryption schema;

— generate AES-256 key (secret key);

— get RSA public key (and certificate);

— cipher secret key with public key, using RSA-OAEP;

— cipher XML element with the embedded file with AES-256 using secret key;

— store all in a file using W3C Encryption schema.

The embedded file is encrypted using a symmetric algorithm (AES-256) with a generated secret key. The Security strength of AES-256 is **256** (NIST SP 800-57 part1).

http://csrc.nist.gov/publications/drafts/800-57/Draft_SP800-57-Part1-Rev3_May2011.pdf

The key transport algorithm RSA-OAEP with mask generation function MGF1 (MGF1p, padding) is used to cipher the generated AES-256 secret key. Key transport algorithms are public key encryption algorithms especially specified for encrypting and decrypting keys . RSA-OAEP uses the receiver's public key to encrypt the secret key generated by AES while encrypting the file. This key transport algorithm chosen is SP800-56B compliant, using KTS-OAEP-basic, without key confirmation.

The AES256 has been chosen for encryption and decryption as the algorithm and key length is safe to use and no security risk is currently know (see NIST SP 800 131 A). Also, RSA is acceptable, with |n|=2048, for SP800-56B key agreement schemas. $|n|$ is the length in bits of the RSA modulus $n = pq$, and $|n| = 2048$ means $|n|$ is at least 2048.

The certificate used to encrypt shall be included in the XML file (as allowed by the W3C encryption schema) to be able to identify the private key corresponding to this certificate (when decrypting).

Basic steps for decryption are:

— read XML document (W3C Encryption schema);

— extract the RSA certificate to ask for (or look for) the corresponding private key;

— decrypt AES secret key using private key;

— decrypt XML element (xenc:CipherValue) with the encrypted content using the secret key;

— as the content of the un-encrypted element should be the file, store this file externally in the file system.

When de-ciphering, the receiver's private key will be used to obtain the AES secret key, which will enable the receiver to decrypt the file using AES256.

5.3.3 File name change upon encryption

When an encryption is applied to a file that has a reserved extended suffix (or, if there is none, a standard suffix), this reserved extended suffix (or, if there is none, a standard suffix) shall change into .encrypted.xml

When an encryption is applied to a file that has no suffix, the reserved extended suffix .encrypted.xml shall be added to the filename.

Table 1 — Encrypted file name examples

File to encrypt	Name of the encrypted file	Filename inside the XML-enc file
Lol	Lol**.encrypted.xml**	Same as « File to sign »

Lol**.pdf**	Lol**.encrypted.xml**	Same as « File to sign »
Lol**.zip**	Lol**.encrypted.xml**	Same as « File to sign »
Lol**.signed.xml**	Lol**.encrypted.xml**	Same as « File to sign »
Lol.**encrypted.xml**	Lol.**encrypted.xml**	Same as « File to sign »

5.3.4 Decrypting a file

Per analogy see previous section

The filename of the decrypted file should become the filename inside the XML signature file.

5.3.5 Signing a file (attached)

The present section explains the requirements and determines the standard finally chosen for applying electronic signatures.

5.3.6 Requirements

The requirements for the choice of the standard were:

- provide non-repudiation: Assure the sender identity, preventing an individual from denying that have effectively signed data;
- prevent the unauthorized (or accidental) modification of data;
- allow the addition of multiple files to a single signature envelope;
- be compliant with European Directive 1999/93/EC;
- use a PKI infrastructure, if required;
- shall contain the signer's digital X.503 v3 certificate;
- shall contain the signing time;
- should include information about policy to verify electronic signature. Hence this signature policy is a legal/contractual document, it's possible that this document is not available for some authorities. The must standard support both situations, in which a regulator has a signature policy or not;
- avoid the use of MD5 or SHA-12);
- long term validation is not needed, as signature should be validated in a limited time-frame.

5.3.7 Electronic signature to use

The file structure generated by the signature shall be XAdES-BES/EPES

http://uri.etsi.org/01903/v1.4.1/

using RSA with SHA512

http://www.w3.org/2001/04/xmldsig-more#rsa-sha512

XAdES-BES/EPES (which has been built up on W3C XML Digital Signature) shall be implemented according to COMMISSION DECISION 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:

http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2011:053:0066:0072:EN:PDF

A signature policy is a legal document that extends the definition of the electronic signature by supplementary properties to respect for signature validation. Depending on the availability of such a signature policy, the file structure to generate shall be:

— XAdES-EPES if an explicit signature policy has been defined by the regulator interested in using with this standard;

— XAdES-BES if no signature policy has been defined by the regulator for use with this standard.

The digital signature for containers will be "SignatureEnveloping", i.e. the output will be an XML file containing as well the signature as the original file. A ds:object element shall contain the Base64 encoding of the file to be signed (if multiple compressed files are needed in the same signature, multiple ds:object elements may be generated). Attributes MimeType, ID, and Encoding shall be included in the ds:object element. ID should be used to store the file-name to enable regeneration of original filename.

The selected signature algorithm for this standard is RSA with SHA-512 as hash function.

The length of the RSA modulus shall be at least 2048 (NIST SP 800-131A-1).

http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf

Details on RSA can be found in RFC 3447 at:

http://www.ietf.org/rfc/rfc3447.txt.

The hash function SHA-512 as specified in FIPS PUB 180-4.

http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

SHA-512 provides a security strength of 256 bits (NIST SP 800-57 part1).

http://csrc.nist.gov/publications/drafts/800-57/Draft_SP800-57-Part1-Rev3_May2011.pdf

5.3.8 File name change upon signature

When a signature is applied to a file that has a reserved extended suffix (or, if there is none, a standard suffix), this reserved extended suffix (or, if there is none, a standard suffix) shall change into .signed.xml

When a signature is applied to a file that has no suffix, the reserved extended suffix .signed.xml shall be added to the filename

Table 2 — Signed file name examples

File to sign	Name of the signed file	Filename inside the XML signature file
Lol	Lol.signed.xml	Same as « File to sign »
Lol**.pdf**	Lol.signed.xml	Same as « File to sign »
Lol**.zip**	Lol.signed.xml	Same as « File to sign »
Lol**.signed.xml**	Lol.signed.xml	Same as « File to sign »

Lol**.encrypted.xml**	Lol.signed.xml	Same as « File to sign »

5.3.9 Validating and extracting a signed file (attached)

Per analogy see previous section.

The filename of the extracted file shall become the filename inside the XML signature file.

5.3.10 Signing a file (detached)

To be added if required

5.3.11 Validating a signed file (detached)

To be added if required

5.3.12 Creating the hash of a file

To be added if required

5.3.13 Verifying the hash of a file

To be added if required

5.4 Creating a submission container

5.4.1 General

In accordance with the requests from EBA and EIOPA, two main characteristics should be given for the Header that is included in every submission container:

— there should be a basic header, which should be small and easy to use;

— the basic Header should be extensible with fields required by the receiver.

These requirements implied a structure of the header as described in the present paragraph.

5.4.2 Header schema structure

The structure of a header as described in this CWA is that of an ExtendedHeader that is to be defined as in Figure 5. Figure 5 shows an Extended Header structure importing the BasicHeader structure that optionally imports the RegisteredOrganisationVocabulary (formerly called « Core Business Vocabularies ») and/or other modules (to be developed in the future).

Figure 5 — Extended Header structure

Table 3 — Characteristics of the XML schemas

Header component	Characteristics
BasicHeader	This header structure is the « smallest possible » header structure. It consists only of an identification of the report (set) as well as of the list of data files composing the submitted report (set). This schema shall be imported into any ExtendedHeader.
ExtendedHeader	This header is an adequate header structure that is to be defined by a receiver (that wants to make use of this CWA) as the header structure to be used by all senders. As an alternative, one of the pre-defined standard headers defined in the next section may be used.
RegisteredOrganizationVocabulary (RegOrg)	From 28th May 2013 the Core Business Vocabulary has been formally published on the W3C standards track as a First Public Working Draft. It has been revised and renamed into Registered Organization Vocabulary (RegOrg). It is described at http://www.w3.org/TR/vocab-regorg/ It's integration into an ExtendedHeader is optional, but it should be imported if any fields defined in the RegisteredOrganizationVocabulary are required in the ExtendedHeader

Every Receiver may define an ExtendedHeader structure in accordance with the own local needs, giving it an adequate namespace.

5.4.3 Predefined standard use-cases of ExtendedHeader schema

The following use-cases for creating an ExtendedHeader are explicitly defined by this CWA and may be used « as is ».

Table 4 — Use-cases for extended headers

ExtendedHeader - pre-defined use-case	Characteristics
BasicHeaderOnly	This header imports the BasicHeader « as is », makes no extensions of it and does not import the RegisteredOrganizationVocabulary as it uses none of its fields. **Namespace:** http://www.eurofiling.info/eu/fr/esrs/Header/BasicHeaderOnly **XSD URL:** http://www.eurofiling.info/eu/fr/esrs/Header/BasicHeaderOnly.xsd **XML sample instance URL:** http://www.eurofiling.info/eu/fr/esrs/Header/BasicHeaderOnly.xml
StandardHeader**With**RegOrg	This header structure reflects the survey made within the Eurofiling BestPractices efforts which had given the results documented in http://www.wikixbrl.info/index.php?title=Best_Practices_on_Common_European_Reporting_Structures All fields related to « Transport » issues have been removed as these are out of scope of this CWA. **Namespace:** http://www.eurofiling.info/eu/fr/esrs/Header/StandardHeader**With**RegOrg **XSD URL:** http://www.eurofiling.info/eu/fr/esrs/Header/StandardHeader**With**RegOrg.xsd **XML sample instance URL:** http://www.eurofiling.info/eu/fr/esrs/Header/StandardHeader**With**RegOrg.xml

StandardHeader**Without** RegOrg	This header is (with regards to its function and its content) equivalent to the previous “StandardHeader**With**RegOrg”, but it does not import RegOrg and creates the missing fields as equivalent simple XML fields **Namespace:** http://www.eurofiling.info/eu/fr/esrs/Header/StandardHeaderWithoutRegOrg **XSD URL:** http://www.eurofiling.info/eu/fr/esrs/Header/StandardHeaderWithoutRegOrg.xsd **XML sample instance URL:** http://www.eurofiling.info/eu/fr/esrs/Header/StandardHeaderWithoutRegOrg.xml

5.4.4 Creating a specific ExtendedHeader schema

The guidelines for the creation of a specific ExtendedHeader schema are given in the external document « How to use the basic header? ».

5.4.5 Creating a header file

The creation of a submission container consists of the actions:

— assembling the data files;

— creation of the header file according to the ExtendedHeader XML schema chosen or defined by the receiver.

5.5 Creating a response container

The creation of a submission container consists of the actions presented in the following sub-clauses.

5.5.1 Creating a ContainerFeedback file

The creation of a ContainerFeedback file should take place in accordance with the documentation of the ContainerFeedback schema in Annex D.

5.5.2 Creating Instance feedback (Validation, usually only for XBRL)

The creation of a InstanceFeedback files should take place in accordance with the documentation of the InstanceFeedback schema in Annex E.

5.6 Security info functions

5.6.1 Digital Certificate and Key generation (example: open source tool)

To be added if required

5.6.2 Digital Certificate and Key distribution

To be added if required

6 Use cases for this CWA

6.1 Introduction

This section will first introduce a general exchange model among a sender and a receiver and then some of the most common use-cases.

6.2 General exchange model between a sender and a receiver

6.2.1 General

The receiver should emit instructions on how to use this CWA for the given exchange of information between a sender and the receiver.

The exchange of information composes of the phases presented in the following sub-clauses.

6.2.2 Phase 1: the sender creates a submission container, applies all security mechanisms required and transmits it to the receiver

The sender makes use of the adequate transport mechanism to submit the container with the data instances.

6.2.3 Phase 2: the receiver processes the security layer(s) on the container and all the files within

The receiver removes all encryption layers and verifies all signatures as indicated by the reserved extended suffixes on the container and the files there-in.

Figure 6 shows an example were two signatures and one encryption have been applied, as well as on all the files within the container

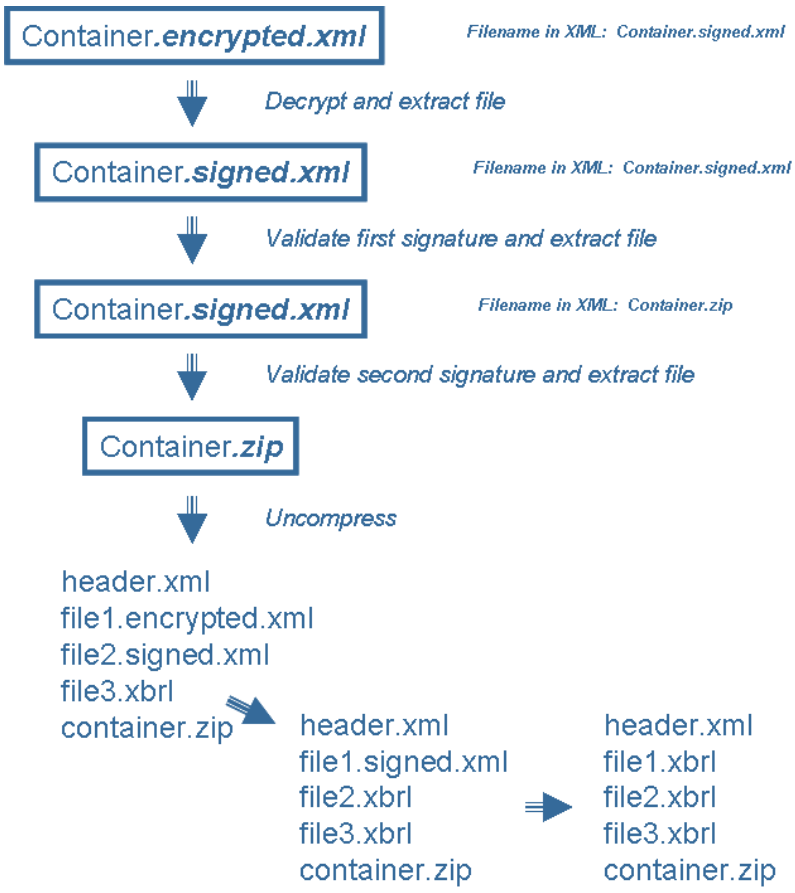

Figure 6 — Security removal at container level

Now the correct structure of the container should be checked (does the header file validate correctly, are all the files announced in the header effectively part of the container, etc.).

6.2.4 Phase 3 (optional): receiver returns a positive / negative acknowledge for the reception of the submission container to the sender

As a result of the processing in the preceding phase, it is now identified if the container could be correctly received or if it was invalid (container or files within not decryptable, signature(s) invalid, entity not known or un-authentifiable, decompression failed, etc.). Now a container feedback file should be generated that either confirms the validity of all reception steps (like security removal and supplementary checks) and results in a positive acknowledge, or that lists (in the opposite case) the errors that occurred (negative acknowledge).

The acknowledgement may either be sent as a separate file to the sender or it may be included in the response container generated in phase 5.

6.2.5 Phase 4: the receiver processes the content of the container

In case of a positive acknowledge, the data files submitted will now go through the next stages of processing

The standard suffixes of the files shall be used to identify those files that can be further processed. As an example, XBRL or XML instances should now be validated by their respective validator (while unstructured data files like word processor files could be made available to analysts for manual review).

As a result of this phase, there should be an adequate feedback file for at least every XBRL instance of the original submission container. A feedback file may either be an instance feedback file as defined by this CWA and / or any other file deemed useful by the receiver to help explaining error conditions of the related submission container.

6.2.6 Phase 5: the receiver returns the validation result for the data files in the response container (optional)

All feedback files should now be added to a response container that will be returned by the receiver to the sender to provide the result of the content processing of a related submission container.

Unlike the submission container, a response container will include no header information.

The receiver may alternatively make available the result of the processing in a way considered more appropriate (e.g. returning links to external systems etc.)

6.3 Container feedback

The container feedback file should be added to the Response container.

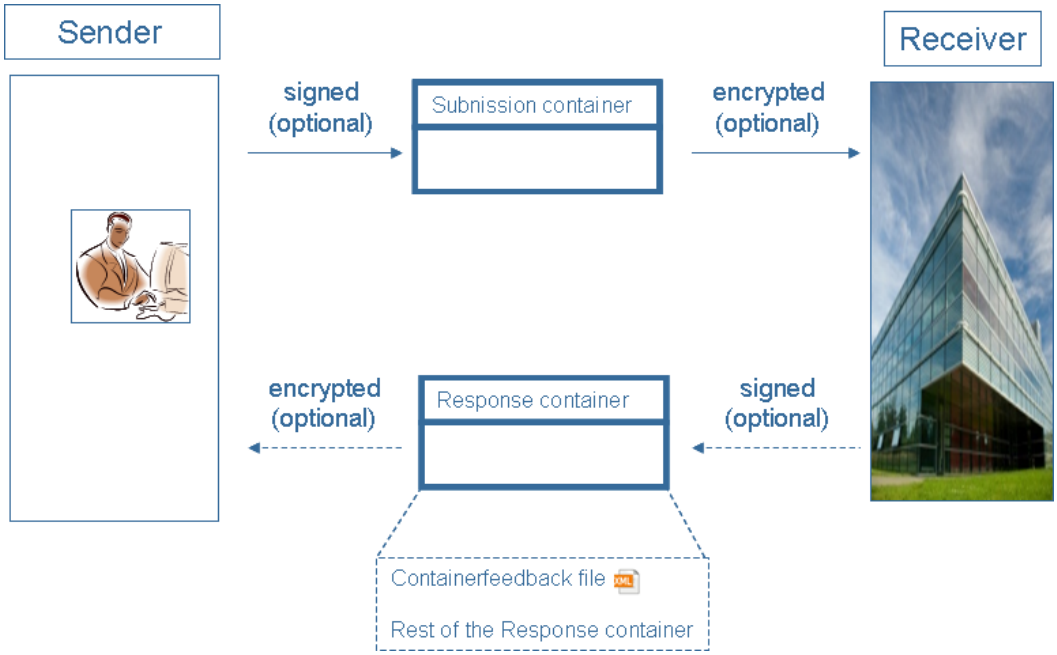

Figure 7 — Simplified exchange model

In case of communication problems among sender and receiver (e.g. exchange of wrong certificates), the content of the ContainerFeedback file may not be accessible for understanding the reasons of the malfunction. As a consequence, only the positive acknowledge is fully reliably accessible, while a negative acknowledge may be well defined in the response container, but still not accessible to the sender.

6.4 Reporting entity to supervisor (1st level)

This use-case is a materialisation of the « General exchange model between a sender and a receiver » ; it is irrelevant which container feedback alternative is chosen. The sender is the reporting entity, the receiver is the supervisor

The security mechanisms applied to submission containers should be the same (and have the same order of application) as those applied to response containers.

6.5 Reporting entity to NSA to ESA (1st and 2nd level)

In this case, the general exchange model is used twice in a row with:

1) General exchange 1: the sender is the reporting entity, the receiver is the NSA;

2) General exchange 2: the sender is the NSA, the receiver is the ESA.

6.5.1 2-layer submission process with forwarding of information

The NSA requires not only data for its own purpose, but also data in a separate container inside the container to be able to forward these data to a subsequent regulator like an ESA. As a consequence, the ESA has all of the reporting entities it is forwarded data from as communication partner s and needs to know their public key / certificate.

Figure 8 shows a 2-layer submission using containers in containers to forward data to subsequent authorities and as well as feedback to the respective sender.

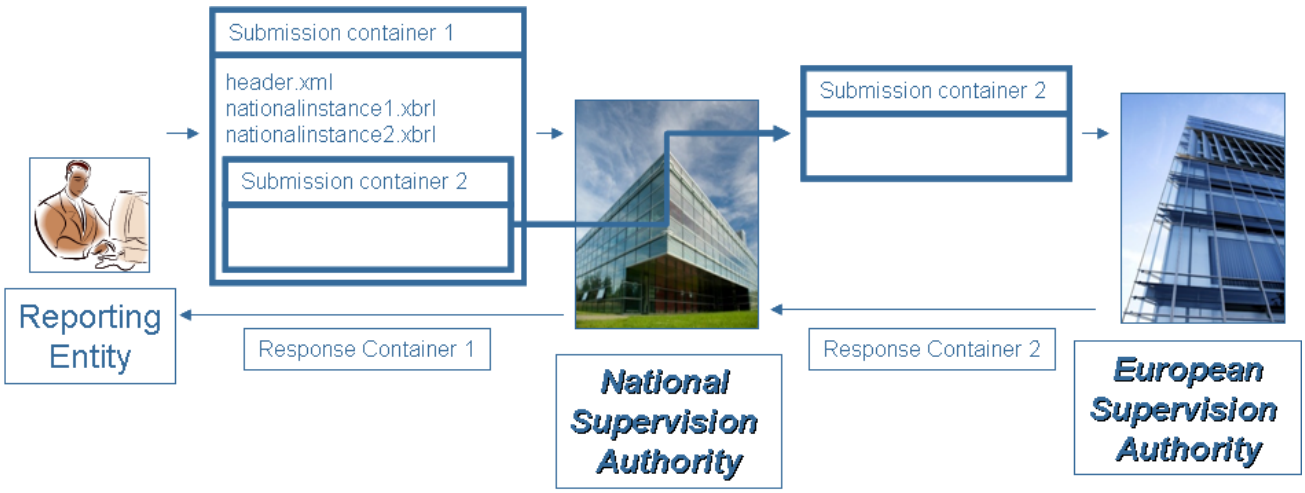

Figure 8 — 2-layer submission process with forwarding of information

6.5.2 2-layer submission process with repackaging

After finishing the submission process from the reporting entity to the NSA, a separate, entirely independent submission process is started using entirely new (reporting entity independent) data prepared by the NSA and submitted to the ESA. The ESA has only one communication partner, the NSA, of which it needs to know the public key / certificate.

Figure 9 shows a 2-layer submission repackaging data into new containers to send them to subsequent authorities

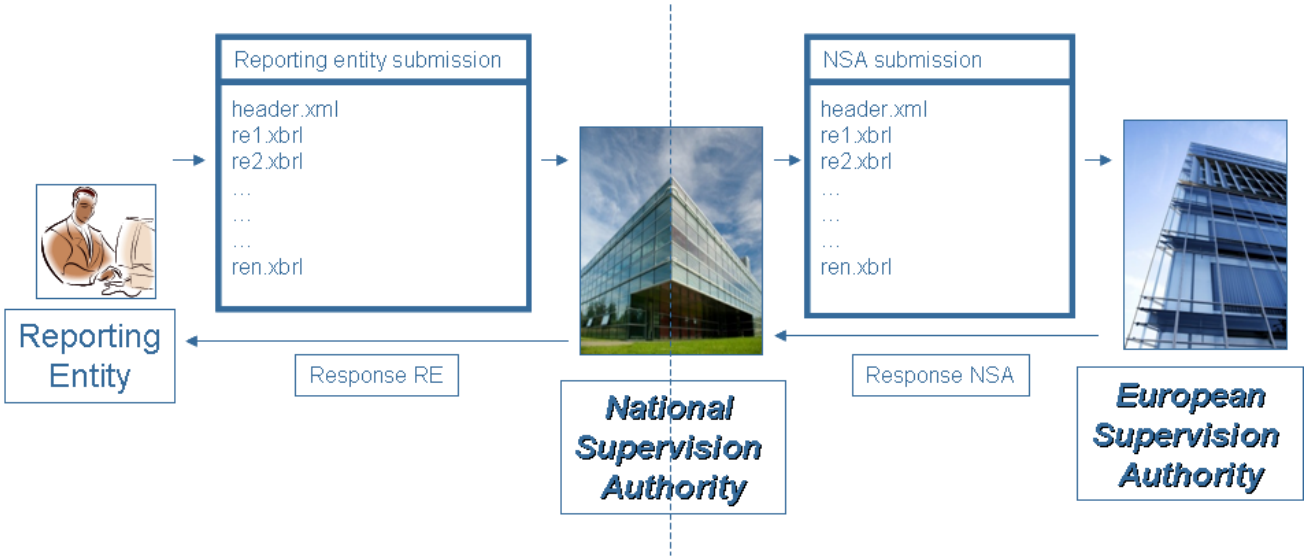

Figure 9 — 2-layer submission process with repackaging

6.5.3 2-layer submission process with regeneration

This corresponds to the case before, but entirely new data will be generated by the NSA.

Figure 10 shows a 2-layer submission, completely new generation of data from NSA systems for subsequent authorities.

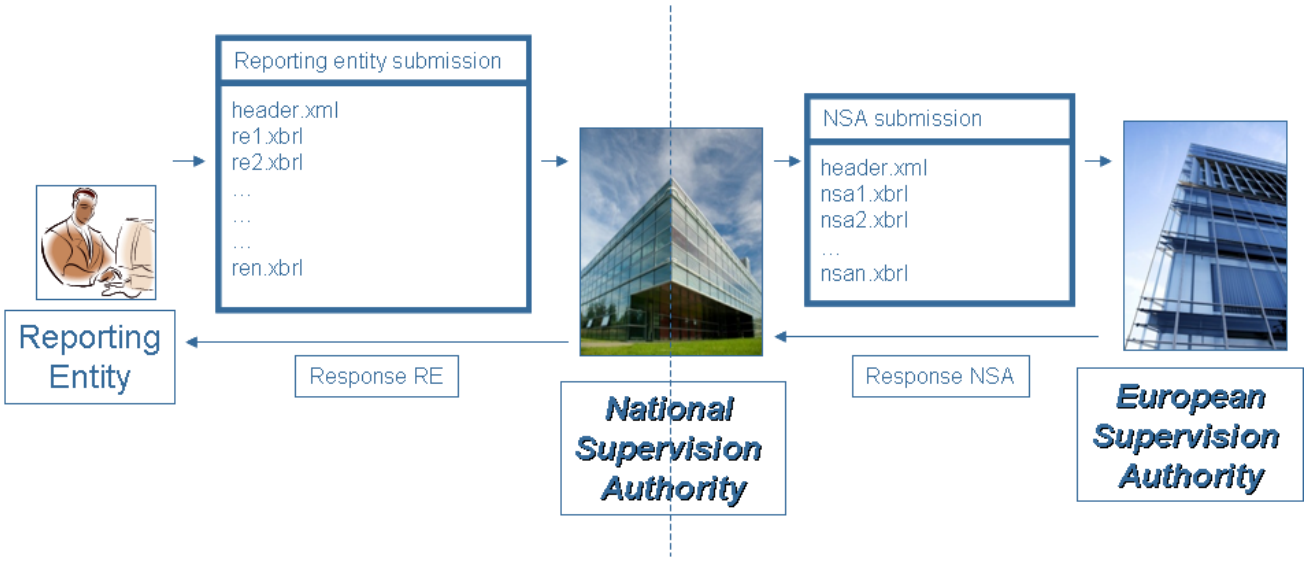

Figure 10 — 2-layer submission process with regeneration

6.6 Extended use case: Web page publication

To be added if required

Annex A
(informative)

Items that shall be defined in the instructions (to be reviewed)

A.1 Introduction

These are some questions to which any institution willing to use this CWA has to give clear answers to in its instructions

A.2 Container structure

Item	Typical instruction	Explanations
Use of CWA encryption layer ?	YES	should be Yes, may be No in environments using a secure transport
Use of CWA signature layer ?	YES	should be Yes, may be No in environments using a secure transport
Differences to the published standard ?	None	List of differences to the standard use of the CWA as published (if applicable)
Allow multiple .zip files per security envelope ?	NO	YES/ NO
Allow update containers ?	NO	YES/ NO
Positive acknowledge of a valid container ?	« Containerfeedback schema »	« None » or « Containerfeedback schema » or « explanation of an alternative mechanism (e.g. links to an external system) »
Use of the « Instancefeedback » schema» to inform of the result of the processing of a data instance ?	« Excel files in folder XBRL_Errors »	« None » or « Instancefeedback schema » or « explanation of an alternative mechanism (e.g. provision of links to an external system of graphical representations of error conditions) »

A.3 Header

These are required precisions on how some tags of the header schema shall be used:

Item	Typical instruction	Explanations
<ReportReferenceID/>	e.g. « Finrep full quarterly consolidated reporting for investment companies » or a code for that reporting	The list of the different reporting identifications covered by the instructions
<AuditStatus>	e.g. « The values « audited » and « not audited » shall be used exclusively »	Either confirm the use of the flag or specify that the value « undetermined » or « in datainstances » is applicable
<ConsolidationStatus>	See <AuditStatus>	See <AuditStatus>
<CapitalCurrency>	Capital Currency shall be used and	This tag may be used for validation purposes

	be EUR mandatorily	in some countries imposing the exclusive use of a single currency
<UpdateStatus>	e.g. « Only use value « Replace » »	If no « update » mechanism is provided, the value « Replace » should be enforced by the instructions, indicating that all prior reports of the same type will be deleted and replaced by the content of the new container In the other case, allowing the values « Update » (keep any values from a prior reporting excepted for those in the data instances of the present container which will be replaced by their new values) and « Delete » (delete any values from prior reportings that are in the data instances of the present container) can make sense
<TestFlag>	« All data are production data, do not use <TestFlag>true</TestFlag> »	This tag may be used to flag data instances sent to the receiver's production infrastructure as test data (to be validated, but not injected into the receiver's databases)

A.4 Lists of codes accepted

A table composed of following information (defining codes accepted both for legal antities as for persons) should explain which codes are allowed:

Code type	Issuer	Country	URI
<IdentifierType>	<IdentifierIssuingAuthority>	<IssuingAuthorityCountry>	<IssuingAuthorityURI>

A list of the applicable Destinees for potential subcontainers should be defined

Annex B
(informative)

Supplementary items that may be useful in the instructions (to be reviewed)

Table B.1 — Supplementary items

Item	Explanations
Name of the use of the CWA	Examples: « Prudential supervisory reporting », « XBRL reporting only », « NSA to EBA transmissions », …
Applicable File naming conventions (if any)	E.g. a link to an external document describing all applicable file naming conventions applicable to containers, folders, data instances, unstructured files
Subcontainers to include	Definition of the subcontainer types, the destinees of these subcontainers, …
Supplementary rules	Example of a supplementary rule: a certain subcontainer shall contain the same instances as the top-level container
Destinees of subcontainers (if applicable)	A list of the applicable Destinees for potential subcontainers should be defined

Annex C
(informative)

Explanations on header schema

This section will be described in detail in the final version of this document. For now, the explanations cover the BasicHeaderOnly use-case:

Table C.5 — BasicHeaderOnly explanations

Item	Explanation
ReportReferenceID	This code identifies the data submitted in the container. It can be a set of reports (e.g. code “FINREP_COREP” for Finrep & Corep), a single report (e.g. “QUARTERLY_CONSOLIDATED_FINREP” for the standard Finrep) or a subset of reports (e.g. “TABLE1&2 FINREP” for only the according subset of Finrep)
<File>/<bh:FilePath>	This field gives the relative URI to a file in the container (starting from top-level)

Annex D
(informative)

Documentation of the container feedback schema

These are 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,

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

— Description: a narrative explanation for the elements.

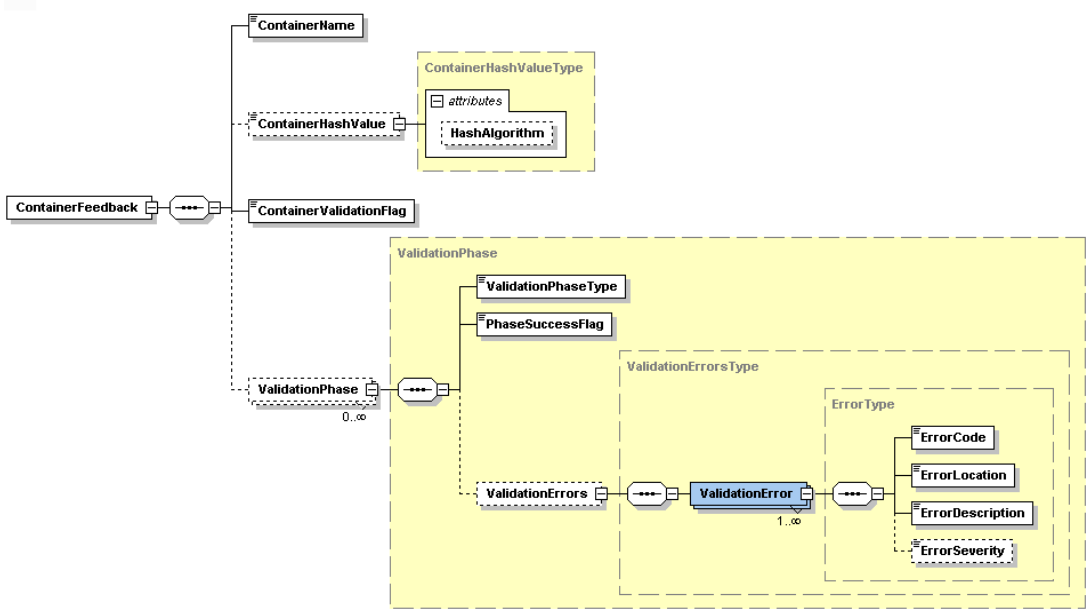

Figure D.1 — Visualisation of the container feedback schema (ContainerFeedback.xsd)

Table D.1 — Container feedback schema element listing and description

Element name	Type	Usage	Description
ContainerFeedback			The root element, validation consists of separate validation sections
ContainerName	text		The name of the container document being receptioned
ContainerHashValue	hash/digest value		Reference to the calculated hash value of the container, it has the " HashAlgorithm " attribute that can be used to indicate which hash algorithm has been used
ContainerValidationFlag	boolean		Overall view if the container structure as a whole was correct true = container ok false = errors found in at least 1 phase of the reception of the container
ValidationPhase	ValidationPhase	Only when ContainerValidationFlag is set "false"	Any phase: decryption, signature verification, authentication, …
ValidationPhase	sequence		
ValidationPhaseType	ValidationPhaseType		The type of validation phase
PhaseSuccessFlag	boolean		true = validation successful false = errors found in the validation phase
ValidationErrors	ValidationErrorsType	Only when PhaseSuccessFlag is set to "false"	lists all the errors found in the validation
ValidationErrorsType	sequence		
ValidationError	ErrorType		The error found
ErrorType	sequence		a generic error type that can be used in all validation sections to define errors found
ErrorCode	identification code		an error code that can be used to identify the error found
ErrorLocation	text		an expression that can be used to locate the error in the instance document, can be an Xpath sentence or line number
ErrorDescription	text		a description of the error found
ErrorSeverity	ErrorSeverityType		The severity of the error
ErrorSeverityType	text enumeration		Possible values: "Info", "Warning", "Error" and "Fatal"
ValidationPhaseType	text enumeration		Different validation phases listed like decryption, signature verification, authentication, …

Annex E
(informative)

Documentation of the instance feedback schema

These are 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,

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

— Description: a narrative explanation for the elements.

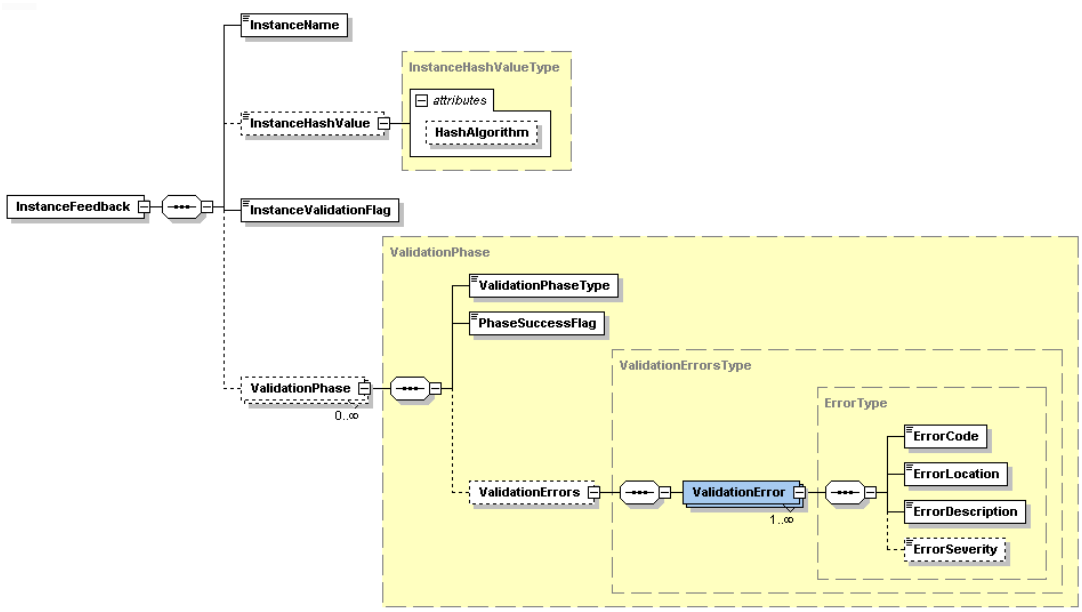

Figure E.1 — Visualisation of the instance feedback schema (InstanceFeedback.xsd)

Table E.1 — Instance feedback schema element listing and description

Element name	Type	Usage	Description
InstanceFeedback			The root element, validation consists of separate validation sections
InstanceName	text		The name of the instance document being validated
InstanceFilePath	hash/digest value		The tag <InstanceFilePath> shall contain the path to the data instance from the top-level of the submission package in relative URI notation
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		Overall view of all istance validations true = all validations successful false = errors found in at least 1 validation phase
ValidationPhase	ValidationPhase	Only when InstanceValidationFlag is set "false"	Any validation: XML, XBRL 2.1 Conformance Suite 1.0 validation, taxonomy validation
ValidationPhase	sequence		
ValidationPhaseType	ValidationPhaseType		The type of validation phase
PhaseSuccessFlag	boolean		true = validation successful false = errors found in the validation phase
ValidationErrors	ValidationErrorsType	Only when PhaseSuccessFlag is set to "false"	lists all the errors found in the validation
ValidationErrorsType	sequence		
ValidationError	ErrorType		The error found
ErrorType	sequence		a generic error type that can be used in all validation sections to define errors found
ErrorCode	identification code		an error code that can be used to identify the error found
ErrorLocation	text		an expression that can be used to locate the error in the instance document, can be an Xpath sentence or line number
ErrorDescription	text		a description of the error found
ErrorSeverity	ErrorSeverityType		The severity of the error
ErrorSeverityType	text enumeration		Possible values: "Info", "Warning", "Error" and "Fatal"
ValidationPhaseType	text enumeration		Different validation phases listed like “XBRL validation”, “XML validation”, …

Bibliography

[1] ETSI Technical Report 102 038 v1.1.1; Electronic Signatures and Infrastructures; XML format for signature policies.

[2] ETSI Technical Specification 101 903 v1.4.1; XML Advanced Electronic Signatures (XAdES).

[3] 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.

[4] CWA 14170; Security requirements for signature creation applications.

[5] CWA 14167-1: Security requirements for trustworthy systems managing certificates for electronic signatures — Part 1: System Security Requirements

[6] 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)

[7] CWA 15579: E-invoices and digital signatures

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

[9] 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.

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

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

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

[13] W3C Recommendation XML Signature Syntax and Processing.

[14] W3C Recommendation XML Encryption Syntax and Processing.

[15] W3C Note XML Encryption Requirements

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

[17] PKWARE Inc., APPnotE.TXT - .ZIP File Format Specification