ETSI TS 119 442 V1.1.1 (2019-02)

**Electronic Signatures and Infrastructures (ESI);
Protocol profiles for trust service providers providing
AdES digital signature validation services**

Reference

DTS/ESI-0019442

Keywords

electronic signature, protocol, validation

Contents

Intellectual Property Rights

Foreword

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

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document specifies the semantics of a protocol for requesting to a remote server (and for receiving the corresponding response) the validation of AdES digital signatures compliant with the following ETSI deliverables: ETSI EN 319 122 [2], ETSI EN 319 132 [3], ETSI EN 319 142 [4], ETSI TS 101 733 [5], ETSI TS 102 778 [9], ETSI TS 101 903 [7], ETSI TS 103 171 [8], ETSI TS 103 172 [10] and ETSI TS 103 173 [6].

The present document specifies the semantics of a second protocol for requesting the augmentation of AdES digital signatures compliant with the aforementioned ETSI deliverables.

The present document also specifies the semantics of a third protocol for requesting the validation and augmentation of AdES digital signatures compliant with the aforementioned ETSI deliverables.

Finally, the present document specifies two bindings, each one in a different syntax (XML and JSON), for each of the aforementioned protocols.

As far as it has been possible and suitable, the protocols have re-used constructs of DSS-X core v2.0: "Digital Signature Service Core Protocols, Elements, and Bindings Version 2.0" [1] (also identified as DSS-X core v2.0 hereinafter). The protocols define new features which are not supported by DSS-X core v2.0.

NOTE 1: The protocols specified in the present document do not include components for submitting to the server ASiC containers compliant with ETSI EN 319 162-1 [i.1], ETSI EN 319 162-2 [i.2], ETSI TS 102 918 [i.3], and ETSI TS 103 174 [i.4]. They do not include either components for reporting on the validation of signatures included within an ASiC container. However, clients can always extract individual signatures and groups of signed documents from ASiC containers and prepare and submit suitable requests to the server for these individual signatures and groups of signed documents.

NOTE 2: The protocols specified in the present document do not include components for submitting to the server time-stamp tokens for their verification. They do not include either components for reporting on the verification of time-stamp tokens. Protocols specified by OASIS DSS-X Technical Committees include this type of components.

NOTE 3: The present document builds on a draft OASIS Committee Specification as the final OASIS specification was not available at the time of publication of the present document. The present deliverable will then be updated when the OASIS Committee Specification is formally adopted.

2 References

2.1 Normative references

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

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

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

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

[1] OASIS Committee Specification Draft 01: "Digital Signature Service Core Protocols, Elements, and Bindings Version 2.0".

NOTE: Available at http://docs.oasis-open.org/dss-x/dss-core/v2.0/csprd01/dss-core-v2.0-csprd01.pdf.

[2] ETSI EN 319 122 (all parts): "Electronic Signatures and Infrastructures (ESI); CAdES digital signatures".

[3] ETSI EN 319 132 (all parts): "Electronic Signatures and Infrastructures (ESI); XAdES digital signatures".

[4] ETSI EN 319 142 (all parts): "Electronic Signatures and Infrastructures (ESI); PAdES digital signatures".

[5] ETSI TS 101 733 (V2.2.1): "Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES)".

[6] ETSI TS 103 173 (V2.2.1): "Electronic Signatures and Infrastructures (ESI); CAdES Baseline Profile".

[7] ETSI TS 101 903 (V1.4.2): "Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES)".

[8] ETSI TS 103 171 (V2.1.1): "Electronic Signatures and Infrastructures (ESI); XAdES Baseline Profile".

[9] ETSI TS 102 778 (all parts): "Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles".

[10] ETSI TS 103 172 (V2.2.2): "Electronic Signatures and Infrastructures (ESI); PAdES Baseline Profile".

[11] ETSI TS 119 102-2 (V1.2.1): "Electronic Signatures and Infrastructures (ESI); Procedures for Creation and Validation of AdES Digital Signatures; Part 2: Signature Validation Report".

[12] IETF RFC 5646: "Tags for Identifying Languages".

[13] ETSI TS 119 102-1 (V1.2.1): "Electronic Signatures and Infrastructures (ESI); Procedures for Creation and Validation of AdES Digital Signatures; Part 1: Creation and Validation".

[14] IETF RFC 3061 (February 2001): "A URN Namespace of Object Identifiers".

[15] H. Andrews. JSON Schema draft 07: "JSON Schema Validation: A Vocabulary for Structural Validation of JSON", March 19, 2018.

NOTE: Available at https://json-schema.org/draft-07/json-schema-validation.html.

2.2 Informative references

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

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

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

[i.1] ETSI EN 319 162-1: "Electronic Signatures and Infrastructures (ESI); Associated Signature Containers (ASiC); Part 1: Building blocks and ASiC baseline containers".

[i.2] ETSI EN 319 162-2: "Electronic Signatures and Infrastructures (ESI); Associated Signature Containers (ASiC); Part 2: Additional ASiC containers".

[i.3] ETSI TS 102 918: "Electronic Signatures and Infrastructures (ESI); Associated Signature Containers (ASiC)".

[i.4] ETSI TS 103 174: "Electronic Signatures and Infrastructures (ESI); ASiC baseline profile".

[i.5] ETSI TS 119 441: "Electronic Signatures and Infrastructures (ESI); Policy requirements for TSP providing signature validation services".

[i.6] ETSI TR 119 001: "Electronic Signatures and Infrastructures (ESI); The framework for standardization of signatures; Definitions and abbreviations".

[i.7] W3C Recommendation (11 April 2013): "XML Signature Syntax and Processing. Version 1.1".

[i.8] ISO 32000-1: "Document management -- Portable document format -- Part 1: PDF 1.7".

3 Definition of terms, symbols, abbreviations and terminology

3.1 Terms

For the purposes of the present document, the terms given in ETSI TR 119 001 [i.6], ETSI TS 119 441 [i.5] and the following apply:

attachment reference container: sub-component of input documents container for transferring a reference to an underlying protocol attachment where either the signed document or the transformed document is placed

augment signature result container: response protocol component for transferring to the client the results of the process carried out by the server when trying to augment one signature

NOTE: This component is specified in clause 6.2.3 of the present document.

augmented signature container: response protocol container for transferring to the client an augmented non-embedded AdES signature

NOTE: The document with signature container and augmented signature container are components of response messages for the augmentation protocol and for the validation and augmentation protocol.

document container: sub-component of input documents container for transferring to the server one signed document or a reference to one underlying transport protocol attachment where the signed document is placed

document digest container: sub-component of input documents container for transferring to the server the digest of one signed document

document with signature container: response protocol container for transferring to the client one signed document embedding its signature(s) or a reference to one underlying transport protocol attachment where the signed document embedding its signature(s) is placed

embedded AdES signature: AdES signature placed within a document that it signs totally or partially

NOTE 1: A XAdES enveloped signature (a XAdES signature that signs a data object that contains the XAdES signature itself) is an example, but there may be other situations where a non enveloped XAdES signature is an embedded XAdES signature, for example a XAdES signature that is a component of a XML file, signs only one specific part of that XML file, and this signed part does not envelope the signature.

NOTE 2: The rationale for this definition is that the placement of the signature to be validated and the signed documents within the protocol messages depends on whether the signature is embedded or not, as specified in clause 5.1.2.

enveloped AdES signature: AdES signature placed within the portion of the document that it signs

NOTE: The portion signed by the signature can be either the whole document or a part of it. What makes the signature be enveloped is that the signature is placed within the signed portion of the document. If the signature is placed within the document but not within the signed portion, then the signature is embedded but not enveloped.

global result component: response protocol component for notifying to the client a generic result of the processing performed by the server following the request submitted by the client

NOTE: If the response contains one or more processing signature results containers this component instructs the client to check the signature processing results. Otherwise, the result has only processed one signature and this component provides the result of this processing.

input documents container: request protocol component for transferring to the server either the signed documents themselves, or the transformed documents, or the digest of the signed documents, or references to underlying transport protocol attachments where the signed documents or the transformed documents are placed

NOTE: For more information about transformations of signed documents, see W3C Recommendation (11 April 2013) [i.7].

representation of a (signed) document: either the (signed) document itself, its digest, or the result of applying to the (signed) document a certain set of known transformations

signature object container: request protocol component for transferring to the server either one non-embedded signature, or a reference to an embedded signature

EXAMPLE: For instance, the client can place a reference to the signature instead of the signature itself in this component when the signature is embedded within the signed document. In these situations, the client can include the signed document (and its embedded signature) within the input documents container and include a reference to the signature within the signature object container.

NOTE 1: From the definitions above, input documents container can contain signatures as long as they are embedded within documents. And signature object container can contain signed documents as long as they are fully enveloped by the signature. The basic principle for placement of signed documents and signatures is the following: a signature that in essence is a non-embedded signature (even if it envelops a signed document) is placed in the signature object container; and an object that in essence is a document (even if it embeds a signature) is placed in or is referenced from the input documents object container.

NOTE 2: Input documents container and signature object container components are implemented by specific XML and JSON types and elements in the bindings defined by the present document.

signature processing results container: response protocol component including the optional outputs generated by the server when processing (validating, augmenting, or validating and augmenting) one specific signature

NOTE: This component is specified in clause 5.2.3.2.1 of the present document. The requests of the three protocols specified in the present document can contain more than one signature. This container includes all the optional outputs generated by the server when it processes one of these signatures. The response message can, consequently, contain one or more signature results containers.

signature reference component: request and response protocol component referencing one signature

NOTE: The XML binding of this component is specified in [11], which is copied in clause 5.1.4.2.4.2 of the present document. Clause 5.1.4.2.4.3 of the present document specifies a JSON binding for this component.

signatures-to-process-refs container: request protocol component that includes references to those signatures whose processing the client requests to the server

NOTE 1: See clause 3.4 of the present document on terminology for an explanation of the meaning of the term "processing".

NOTE 2: A request message can include more than one signature. This component allows the client to instruct the server to process (validate, augment, or validate and augment) a selected subset of them.

transformed document container: sub-component of input documents container for transferring to the server the transformed document or a reference to one underlying transport protocol attachment where the transformed document is placed

3.2 Symbols

Void.

3.3 Abbreviations

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

CMS	Cryptographic Message Syntax
DSS-X	Digital Signature Services eXtended
DSSX, DSS-X	Digital Signature Services eXtended
ERS	Evidence Record Syntax
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
OASIS	Organization for the Advancement of Structured Information Standards
OCSP	Online Certificate Status Protocol
OID	Object Identifier
RFC	Request For Comments
URI	Uniform Resource Identifier
URN	Uniform Resource Name
UTC	Universal Time Coordinated
XML	eXtensible Markup Language

3.4 Terminology

The term "digest of the document", "document" being one of the documents signed by one signature submitted to a server within the request of any of the three protocols defined in the present document, is understood as indicated below:

- If the document is signed by a CAdES signature, it is the digest of the signed document itself.
- If the document is signed by a XAdES signature, it is the digest computed as specified inW3C Recommendation (11 April 2013) [i.7].
- If the document is a PDF document signed by a PAdES signature built on CMS or CAdES, it is the digest computed as specified in ISO 32000-1 [i.8].

The term "process" applied to AdES signature means either "validate", or "augment", or "validate and augment" depending of the protocol where the term is used. If the term is used out of the context of one protocol it does mean the action performed by the server on the signature, which is one of the three actions aforementioned.

4 Technical approach to the specification of the protocols

4.1 Main features

The main features supported by the 'validation' protocol specified in the present document, which are not supported by DSS-X core v2.0 [1] are:

1) Supports requesting the validation of one or more PAdES signatures embedded in one PDF document, if the client submits it to the server. It supports the validation of one PAdES signature if the client only submits the digest of the document where the aforementioned signature is placed.

2) Supports, when the request message contains more than one signature, requesting the validation of a subset of them.

3) Supports requesting to the server the application of a certain signature validation policy for validating the AdES signature(s). The server may notify in the response, the signature validation policy applied. The server may notify the list of signature validation policies that it supports.

4) Supports requesting a signed or unsigned detailed validation report for each validated signature. The server may include one signed or unsigned validation report for each signature within the response. The server may also include one signed or unsigned validation report for several validated signatures.

NOTE: The ETSI TS 119 102-2 [11] defines a validation report that can contain details of the validation of one or more AdES signatures.

5) The server may generate one or more signature processing results containers, each one providing all the details (including the aforementioned signed or unsigned validation report) concerning the validation of one signature.

The main features supported by the 'validation and augmentation' protocol specified in the present document, which are not supported by DSS-X core v2.0 [1] are:

1) Supports requesting the validation and augmentation of one or more PAdES signatures embedded in one PDF document, if the client submits it to the server. It supports the validation and augmentation of one PAdES signature if the client only submits the digest of the document where the aforementioned signature is placed.

2) Supports, when the request message contains more than one signature, requesting the validation and augmentation of a subset of them.

3) Supports requesting to the server the application of a certain signature validation policy for validating the AdES signature(s). The server may notify in the response, the signature validation policy applied. The server may notify the list of signature validation policies that it supports.

4) Supports requesting a signed or unsigned detailed validation report for each validated signature. The server may include one signed or unsigned validation report for each signature within the response. The server may also include one signed or unsigned validation report for several validated signatures.

5) The server may generate one or more signature processing results containers, each one providing all the details (including the aforementioned signed or unsigned validation report) concerning the validation and the augmentation of one signature.

The main features supported by the 'augmentation' protocol specified in the present document are:

1) Supports requesting the augmentation of one or more XAdES signatures and of one or more of CAdES signatures. The incorporation of the signatures and the signed documents are as specified in DSS-X core v2.0 [1].

2) Supports requesting the augmentation of one or more PAdES signatures embedded in one PDF document, if the client submits it to the server. It supports the augmentation of one PAdES signature if the client only submits the digest of the document.

3) Supports, when the request message contains more than one signature, requesting the augmentation of a subset of them.

4) Supports submission of validation material required for augmenting the signatures.

5) Supports submission of the claimed identity of the client.

6) The server may generate one or more signature processing results containers, each one providing all the details concerning the augmentation of one signature.

4.2 General requirements

The protocols specified in the present document re-use, wherever it is possible, the components specified in DSS-X core v2.0 [1].

Wherever the protocols defined in the present document require a certain component whose semantics and/or syntax is not offered by any of the components specified in DSS-X core v2.0 [1], the present document defines and fully specifies new components.

For components re-used from DSS-X core v2.0 [1], and in the absence of any further requirement defined in the present document, the requirements defined in DSS-X core v2.0 [1], shall apply. The present document may define additional requirements for these re-used components. In case that a requirement defined in the present document contradicts any requirement defined in DSS-X core v2.0 [1] the requirement defined in the present document shall take precedence.

For components re-used from DSS-X core v2.0 [1], and in the absence of a different processing model defined in the present document, the processing model (including results returned by the server) defined in DSS-X core v2.0 [1] shall apply. The present document may modify the processing model defined in DSS-X core v2.0 [1] for these components. In case that a certain aspect of the processing model defined in the present document contradicts any aspect of the processing model defined in DSS-X core v2.0 [1], the processing model defined in the present document shall take precedence.

EXAMPLE: The present document does not specify, for instance, that the presence of a certain re-used optional component in the request message implies the presence of a certain re-used optional component in the response message: this is a requirement inherited from the aforementioned OASIS specifications.

NOTE: The protocols defined in the present document on one hand restrict the degree of optionality for certain features of the verification protocol defined in DSS-X core v2.0 [1]. But at the same time, on the other hand, some of the protocols defined in the present document incorporate features that are not present in DSS-X core v2.0 [1].

Services implementing the protocols defined in the present document shall support all the components specified in the present document, which apply to the AdES signature type(s) that the service is able to process (i.e. to validate, to augment, or to validate-and-augment), regardless whether their incorporation in the messages is mandatory or optional.

The rest of the present document is organized as follows:

1) Clauses 4.3 and 4.4 provide general remarks on the XML and JSON protocols relying on the DSS-X core v2.0 protocol.

2) Clause 5 specifies all the components for the validation protocol in its two bindings (XML and JSON) relying on the DSS-X core v2.0 protocol.

3) Clause 6 specifies those specific components for the augmentation protocol in its two bindings (XML and JSON) relying on the DSS-X core v2.0 protocol.

4) Clause 7 specifies those specific components for the validation and augmentation protocol in its two bindings (XML and JSON) relying on the DSS-X core v2.0 protocol.

5) Clause 8 specifies the processing models for the three protocols.

6) Clause 9 specifies components for allowing asynchronous processing in the three protocols.

For each component of the aforementioned protocols, the present document:

1) Defines requirements for the semantics of the component (i.e. its mandatory contents, its optional contents, etc.). These requirements are defined in clauses "Component semantics".

2) Defines requirements for the components of the XML binding within clauses named "XML component".

3) Defines requirements for the JSON component of the JSON binding within clauses named "JSON component".

4.3 XML protocol

4.3.1 Introduction

The new structures defined in the present document are contained in the XML schema file "19442xmlSchema.xsd". This file also contains the redefinition of certain structures defined in DSS-X core v2.0 [1], as shown in clause 4.3.2 of the present document. The new elements and types defined in that schema are defined within the XML namespace whose URI value is shown below:

- `http://uri.etsi.org/19442/v1.1.1#`

Table 1 shows the URI values of other XML namespaces and their corresponding prefixes used in the aforementioned schema file and within the present document.

Table 1

URI value of the XML Namespace	Prefix
http://uri.etsi.org/19442/v1.1.1#	etsival
http://uri.etsi.org/19102/v1.2.1#	etsivr
http://docs.oasis-open.org/dss/ns/core	dss2
http://docs.oasis-open.org/dss/ns/base	dsb
http://www.w3.org/2000/09/xmldsig#	ds
urn:oasis:names:tc:SAML:2.0:assertion	saml2
http://www.w3.org/2001/XMLSchema	xs

4.3.2 Redefined DSS-X types

The present document redefines three types already defined by DSS-X core v2.0 [1]. These redefinitions are contained in the XML schema file "19442xmlSchema.xsd" and are copied below for information, as well as in the clauses where these types are specified:

```
<!—targetNamespace="http://uri.etsi.org/19442/v1.1.1#" -->

<xs:redefine schemaLocation="foo_dss_core.xsd">
    <xs:complexType name="dss2:OptionalInputsVerifyType">
        <xs:complexContent>
            <xs:extension base="dss2:OptionalInputsVerifyType">
                <xs:sequence>
                    <xs:element ref="etsival:ProcessSignatures" minOccurs="0"/>
                    <xs:element ref="etsival:UseSignatureValidationPolicy" minOccurs="0"/>
                    <xs:element ref="etsival:ReturnValidationReport" minOccurs="0"/>
                    <xs:element ref="etsival:ReturnAugmentedSignature" minOccurs="0"/>
                    <xs:element ref="etsival:ProofsOfExistence" minOccurs="0"/>
                    <xs:element name="TSTokensQualityLevel" type="xs:anyURI" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="dss2:OptionalOutputsVerifyType">
        <xs:complexContent>
            <xs:extension base="dss2:OptionalOutputsVerifyType">
                <xs:sequence>
                    <xs:element ref="etsival:ValidationReport" minOccurs="0"/>
                    <xs:element ref="etsival:ResultsForOneSignature" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element ref="etsival:AppliedSignatureValidationPolicy" minOccurs="0"/>
                    <xs:element ref="etsival:AvailableSignatureValidationPolicies" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="dss2:DocumentHashType">
        <xs:complexContent>
        <xs:extension base="dss2:DocumentHashType">
            <xs:sequence>
                <xs:element name="PAdESFieldName" type="xs:string" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:redefine>
```

- Clause 5.1.3.2.2.3 specifies the redefined `dss2:DocumentHashType`.
- Clause 5.1.4.1.2 specifies the redefined `dss2:OptionalInputsVerifyType`.
- Clause 5.2.3.1.2 specifies the redefined `dss2:OptionalOutputsVerifyType`.

NOTE: The present document redefines `dss2:DocumentHashType` because this component has to have an additional component for containing a PDF field name. This is required for allowing to submit the digest document when the client requires validation of PAdES signatures.

4.4 JSON protocol

4.4.1 Introduction

The new structures defined in the present document are contained in the schema file "19442jsonSchema.json". This file also contains the extensions of certain structures defined in DSS-X core v2.0 [1]. These structures are listed in clause 4.4.2 of the present document.

4.4.2 Extension of DSS-X types

The present document extends three types defined by DSS-X core v2.0 [1]. These extensions are contained in the JSON schema file "19442jsonSchema.json".

The extension mechanism is implemented using the keyword "`allOf`" as specified in "JSON Schema Validation: A Vocabulary for Structural Validation of JSON" [15].

The types that are defined as extensions of DSS-X types are listed below:

- `OptionalInputsVerifyType` extends `dss2-OptionalInputsVerifyType`. It is defined in clause 5.1.4.1.3.
- `OptionalOutputsVerifyType` extends `dss2-OptionalOutputsVerifyType`. It is defined in clause 5.2.3.1.3.
- `DocumentHashType` extends `dss2-DocumentHashType`. It is defined in clause 5.1.3.3.2.3.

NOTE: The present document defines this new type extending `dss2-DocumentHashType` because this component has to have an additional component for containing a PDF field name. This is required for allowing to submit the digest document when the client requires validation of PAdES signatures.

The present document defines two additional new types that are built on some of the aforementioned types, namely:

- `InputDocumentsType`, that builds on `DocumentHashType`. It is defined in clause 5.1.3.3.1.
- `VerifyRequestType`, that builds on `OptionalInputsType` and `InputDocumentsType`. It is defined in clause 5.1.1.3.

5 Protocol for validation of AdES signatures

5.1 Request message

5.1.1 Component for requesting validation

5.1.1.1 Component semantics

The message for requesting the validation of an AdES signature to a remote server shall contain components for:

1) Submitting either the signature to be validated or a reference to the signature to be validated when this signature is enveloped within a signed document. Clause 5.1.2 specifies semantic requirements for this component.

2) Submitting the signed document(s) or representation(s) of these signed document(s). Clause 5.1.3 specifies semantic requirements for this component.

NOTE 1: When the signature to validate is separate from all or part of the documents it signs, the signature is placed in one component and the signed document(s) is (are) placed in another component.

NOTE 2: When the signature to validate envelops the signed document (this is the case of a XAdES enveloping signature that is not embedded in other document -a XAdES signature can envelop one document and be embedded in another document- or a CAdES attached structure) or it is placed within the signed document (this is the case of a PDF document with one or more PAdES signatures, or an embedded XAdES signature), signature and signed documents are placed in one component.

3) Identifying one or more protocols and/or profiles that the request message is compliant with. The first one of such components shall have the following URI as value, identifying the request message as one that has been built using the "validation" protocol specified in the present document:

   - `http://uri.etsi.org/19442/v1.1.1/validationprotocol#`.

This message may contain a unique identifier.

This message may contain an identifier of the service policy the client requests the server to use for processing the signatures.

This message may contain other components for requesting to the server additional features. Clause 5.1.4.1 lists these optional components and contains references to clauses that specify semantic requirements for each component.

5.1.1.2 XML component

The element that shall be the component for requesting the validation of AdES signature(s) shall be the element `dss2:VerifyRequest` as specified in clause 4.2.10.2 of DSS-X core v2.0 [1].

The `dss2:VerifyRequest` element shall be as specified in DSS-X core v2.0 [1].

The `dss2:VerifyRequest` element shall have one or more `dsb:Profile` children elements. The first one shall have the value `http://uri.etsi.org/19442/v1.1.1/validationprotocol#`, identifying the request as a validation request compliant with the validation protocol specified in the present document.

The `dss2:SignatureObject` child element shall not contain any time-stamp token.

Any optional component specified in clause 5.1.4 shall appear as child of the `dss2:OptionalInputs` child element of the `dss2:VerifyRequest` element.

5.1.1.3 JSON component

The element that shall implement the verify request message for the JSON binding of the protocol shall be the `VerifyReq` element. This element shall be an instance of the `VerifyRequestType`, defined as in JSON Schema file "19442jsonSchema.json", whose location is detailed in clause A.1, and is copied below for information.

```
"VerifyRequestType": {
  "type": "object",
  "properties": {
    "profile": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "reqID": {
      "type": "string"
    },
    "inDocs": {
      "$ref": "#/definitions/InputDocumentsType"
    },
    "optInp": {
      "$ref": "#/definitions/OptionalInputsVerifyType"
    },
    "sigObj": {
      "$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-SignatureObjectType"
    }
  }
}
```

The profile, `reqID`, and `sigObj` elements shall be as specified in clauses 4.2.10.1, and 4.2.8.1 of DSS-X core v2.0 [1].

The profile array shall have one or more items as specified in clause 4.2.10.1 of DSS-X core v2.0 [1]. The first one shall have the value `http://uri.etsi.org/19442/v1.1.1/validationprotocol#`, identifying the request as a validation request compliant with the validation protocol specified in the present document.

The `optInp` shall be as specified in clause 5.1.4.1.3 of the present document.

The `inDocs` element shall be as specified in clause 5.1.3.3 of the present document.

5.1.2 Component for submitting signature to be validated

5.1.2.1 Component semantics

The protocol shall allow including the signature in different containers according to the following rules:

1) If the signature is an embedded AdES signature, the embedding document and the signature shall be placed either:

   - within one sub-component of the input documents container; or

   - within an underlying protocol attachment. In this case, a sub-component of the input documents container shall include a reference to the aforementioned attachment.

   Additionally the signature object container shall contain a reference to the embedded signature.

2) If the signature is a non-embedded AdES signature, it shall be placed within the signature object container.

3) If the signature is embedded AdES signature and the digest of the embedding document is submitted instead the document itself, then the AdES signature shall be placed within the signature object container.

NOTE: This implies that the protocols defined in the present document support submitting several PAdES signatures embedded within one PDF document, if this one is submitted to the server. However, if the client submits the digest of the document, the request message can only have one PAdES signature.

Table 2 shows the cardinalities of the components required in this protocol for incorporating signature(s), for requesting the validation of AdES signatures, depending on its types (CAdES, PAdES or XAdES) and their relative position with respect the signed document(s).

Rows in the table show information corresponding to the different types of AdES signatures whose validation is requested, as well as their relative position to the signed document(s).

The first column shows the different types of AdES signatures that can be submitted to the server.

Column I shows the cardinalities of the signature object container.

Colum II shows the contents of the signature object container. The values appearing in the cells of this column may be the following ones:

- Signature: This value indicates that the signature object container contains the non-embedded signature itself.
- Reference: This content appears when submitting an embedded signature. It indicates that the signature object container shall contain a reference to the embedded signature.

Column III shows the cardinalities of the input documents container that includes either a document embedding the signature or a reference to an underlying protocol attachment that contains the document embedding the signature.

Each cell in I and III columns indicates the required cardinality of the component shown in the header of the corresponding column, for the type of signature, located in a relative position to the signed document(s) as indicated in the header of the corresponding row. An integer value indicates an exact number of components, "*" stands for "0 or more", and "0..1" means "0 or 1".

Table 2: Placement of signatures

Component / **Type of signature**	**I** **Signature object container**	**II** **Contents of signature object container**	**III** **Input documents container containing embedded signatures or references to attachments with signatures**
CMS structure attached	1	Signature	0
CMS structure detached	1	Signature	0
Non-embedded XAdES	1	Signature	0
Embedded XAdES	0..1	Reference	1
PAdES enveloped within the PDF document (embedded signature)	0..1	Reference	1
NOTE: Within one CMS structure there are as many CAdES signatures as items within the `signerInfos` component.			

5.1.2.2 XML components

The element that shall be the component for submitting the signature(s) to be validated shall be either:

a) the `dss2:SignatureObject` child element of the `dss2:VerifyRequest` root element if the signature is not embedded within the signed document(s); or

b) the `dss2:InputDocument` child element of the `dss2:VerifyRequest` element if the signature(s) is embedded within the signed document.

The `dss2:SignatureObject` element shall be the XML implementation of the signature object container and the `dss2:InputDocument` element shall be the XML implementation of the input documents container within the XML binding of the protocol.

The requirements governing the presence, cardinalities, and contents of the aforementioned elements are given in clause 5.1.3.4.

5.1.2.3 JSON component

The element that shall be the component for submitting the signature(s) to be validated shall be either:

a) the `sigObj` child element of the `VerifyReq` root element if the signature is not embedded within the signed document(s); or

b) the `inDocs` child element of the `VerifyReq` element if the signature(s) is embedded within the signed document.

The requirements governing the presence, cardinalities, and contents of the aforementioned elements are given in clause 5.1.3.4.

5.1.3 Components for submitting signed documents or representations of the signed documents

5.1.3.1 Components semantics

The protocol shall allow including the signed document in several containers depending on its relative position regarding the signature that signs it:

1) If a non-embedded signature envelops the document, then the client shall place the enveloping signature and the enveloped signed document within the signature object container.

2) If the signature is not as in item 1), then the client shall place the signed document either directly within the input documents container or within an underlying protocol attachment. Additionally, the input documents container shall contain one sub-component containing either the document or a reference to the attachment where the document is placed.

The current protocol shall also allow submitting to the server other representations of the signed documents different than the actual documents. Each type of representation shall be placed in a different sub-component of the input documents container, as indicated below:

a) If a client wants to submit the digest of the actual signed document, then the client shall place this digest within the document digest container.

b) If a client wants to submit the result of transforming a document, it shall place this transformed document either:

- within the transformed document container; OR
- within an underlying transport protocol attachment. In this case, the transformed document container shall include a reference to this attachment.

Table 3 shows the cardinalities of the components required in this protocol for incorporating signed documents, or signed documents representations (transformed documents and documents digests) for requesting the validation of AdES signatures, depending on the types of the signatures and their relative position with the signed document(s).

Rows in the table show information corresponding to the different types of AdES signatures whose validation is requested, as well as their relative position to the signed document(s).

The first column shows the different types of AdES signatures that can be submitted to the server.

Column **Document container** shows the cardinalities of the document container.

Column **Transformed document container** shows the cardinalities of transformed document container.

Column **Document digest container** shows the cardinalities of document digest container.

The conventions used in Table 3 for indicating cardinalities are the same as the conventions used in Table 2.

Table 3: Components containing either documents or representations of documents in validation requests messages

Component and subcomponents / Type of signature	Input documents container containing documents or documents representations		
	Document container	**Transformed document container**	**Document digest container**
CMS structure attached	0	0	0
CMS structure detached	0..1	0	0..1
Non-embedded XAdES	*	*	*
Embedded XAdES	1..*	*	*
PAdES enveloped within the PDF document	0..1	0	0..1
NOTE 1: In the case of embedded XAdES document container has a minimum cardinality of 1, as it is the only way of submitting to the server the embedded signature: within the signed document that embeds it. If the signature is not embedded, then either the document or a representation of the document can be submitted as the signature is submitted within the signature object container. NOTE 2: The protocols defined in the present document allow sending either one PDF document enclosing one or more PAdES signatures, or one PAdES signature detached from any PDF document and the digest of the document, understood as specified in clause 3.4. As that clause mentioned, details on how to compute the digest for each signature of this type, can be found in ISO 32000-1 [i.8].			

5.1.3.2 XML components

5.1.3.2.1 General requirements

The `dss2:SignatureObject` element shall implement the signature object container for the XML binding of the protocol.

The `dss2:InputDocuments` element shall implement the input documents container for the XML binding of the protocol.

The requirements governing the presence, cardinalities, and contents of the XML components are given in clause 5.1.3.4.

5.1.3.2.2 Additional requirements for contents of `dss2:InputDocuments`

5.1.3.2.2.1 Element `dss2:Document` for sending original documents

The `dss2:Document` element shall implement the document container for the XML binding of the protocol.

NOTE: `dss2:Base64Data` child element of the `dss2:Document` can contain either the signed document in its `dsb:Value` child element, or a reference to an underlying protocol attachment (where the signed document is placed) in its `dsb:AttRef` child element.

5.1.3.2.2.2 Element `dss2:TransformedData` for sending transformed documents

The `dss2:TransformedData` element shall implement the transformed document container for the XML binding of the protocol.

If the signature(s) to be validated is(are) XAdES signature(s), and if the client wants to submit to the server not the original document, but the result of a set of transformations applied to it, then the client shall incorporate the base-64 encoding of the binary representation of the result of applying this set (which may be either the full sequence of transformations specified in their `ds:Reference` ancestor element, or a part of it) to the original document either into the `dss2:Base64Data` child element of the `dss2:TransformedData` child element of the `dss2:InputDocuments` or within an underlaying protocol attachment.

The client shall submit one `dss2:TransformedData` element for each result of applying a sequence of transformations to one of the original documents.

The `dss2:TransformedData` element shall incorporate the `WhichReference` attribute.

NOTE: `dss2:Base64Data` child element of the `dss2:TransformedData` can contain either the transformed document in its `dsb:Value` child element, or a reference to an underlying protocol attachment (where the transformed document is placed) in its `dsb:AttRef` child element.

5.1.3.2.2.3 Element `dss2:DocumentHash` for sending digest of documents

The element that shall implement the digest document container for the XML binding of the protocol shall be the `dss2:DocumentHash` element, instance of `dss2:DocumentHashType`, redefined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1. The redefinition of `dss2:DocumentHashType` has been copied in clause 4.3.2 for information.

The requirements for each instance of the redefined the `dss2:DocumentHashType` shall be as specified in clause 4.2.5.1 of DSS-X core v2.0 [1].

In addition, the following requirements shall also apply:

- If the signature(s) to be validated is(are) XAdES then the `dss2:DocumentHash` element shall incorporate the `WhichReference` attribute and shall not incorporate the `PAdESFieldName` element.
- If the signature(s) to be validated is(are) CAdES or PAdES built on CMS then the `dss2:DocumentHash` element shall not incorporate the `WhichReference` attribute.
- If the signature(s) to be validated is(are) PAdES then the `PAdESFieldName` element shall be present and it shall have as value the name of the PDF field where the PAdES signature is placed within the PDF signed document.

5.1.3.3 JSON components

5.1.3.3.1 General requirements

The `sigObj` element shall implement the signature object container for the JSON binding of the protocol.

The `inDocs` element shall implement the input documents container for the JSON binding of the protocol. It shall be an instance of the `InputDocumentsType`, defined as in JSON Schema file "19442jsonSchema.json", whose location is detailed in clause A.1, and is copied below for information.

```
"InputDocumentsType": {
  "type": "object",
  "properties": {
    "doc": {
      "type": "array",
      "items": {
        "$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-DocumentType"
      }
    },
    "transformed": {
      "type": "array",
      "items": {
        "$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-TransformedDataType"
      }
    },
    "docHash": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/DocumentHashType"
      }
    }
  }
}
```

The requirements governing the presence, cardinalities, and contents of the JSON elements are given in clause 5.1.3.4.

The component `doc` shall be as specified in clause 4.2.3.1 of DSS-X core v2.0 [1].

The component `transformed` shall be as specified in clause 4.2.4.1 of DSS-X core v2.0 [1].

The component `docHash` shall be as specified in clause 5.1.3.3.2.3 of the present document.

5.1.3.3.2 Additional requirements for contents of `inDocs`

5.1.3.3.2.1 Element `doc` for sending original documents

Each item within the `doc` array shall implement a document container for the JSON binding of the protocol.

NOTE: `b64Data` child element of one item within the `doc` array can contain either the signed document in its `val` child element, or a reference to an underlying protocol attachment (where the signed document is placed) in its `attRef` child element.

5.1.3.3.2.2 Element `transformed` for sending transformed documents

Each item within the `transformed` array shall implement a transformed document container for the JSON binding of the protocol.

If the signature(s) to be validated is(are) XAdES signature(s), and if the client wants to submit to the server not the original document, but the result of a set of transformations applied to it, then the client shall incorporate the base-64 encoding of the binary representation of the result of applying this set (which may be either the full sequence of transformations specified in their `ds:Reference` ancestor element, or a part of it) to the original document either into the `base64Data` child element of the `transformed` child element of the `inDocs` or within an underlaying protocol attachment.

The client shall submit one `transformed` element for each result of applying a sequence of transformations to one of the original documents.

The `transformed` element shall incorporate the `whichRef` element child.

NOTE: `b64Data` component of an item within the `transformed` array can contain either the transformed document in its `val` child, or a reference to an underlying protocol attachment (where the transformed document is placed) in its `attRef` child element.

5.1.3.3.2.3 Element `docHash` for sending digest of documents

The element that shall implement one digest document container for the JSON binding of the protocol shall be an instance of the `DocumentHashType` , resulting from extending `dss2-DocumentHashType`, defined as in JSON Schema file "19442jsonSchema.json", whose location is detailed in clause A.1, and is copied below for information.

```
"DocumentHashType": {
  "type": "object",
  "allOf": [
    {"$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-DocumentHashType"}
  ],
  "properties": {
    "pAdESSFieldName": {"type": "string"}
  }
}
```

The requirements for each instance of the `DocumentHashType` shall be as specified in clause 4.2.5.1 of DSS-X core v2.0 [1].

In addition, the following requirements shall also apply:

- Each instance of this type shall contain the `di` property.
- If the signature(s) to be validated is(are) XAdES then the instance of the extended `DocumentHashType` shall incorporate the `whichRef` element and shall not incorporate the `pAdESSFieldName` element.
- If the signature(s) to be validated is(are) CAdES or PAdES built on CMS then the instance of the extended `DocumentHashType` shall not incorporate the `whichRef` element.

- If the signature(s) to be validated is(are) PAdES then the instance of the extended `DocumentHashType` shall include the `pAdESFieldName` element. This element shall have as value the name of the PDF field where the PAdES signature is placed within the PDF signed document.

5.1.3.4 Cardinalities for elements used for sending signatures, signed documents and representations of signed documents

Table 4 shows the cardinalities of the XML and JSON elements required in this protocol for incorporating signature(s), signed documents, and signed documents representations (transformed documents and documents digests) for requesting the validation of AdES signatures, depending on the types of the signatures and their relative position with the signed document(s).

Columns in the table show information corresponding to the different types of AdES signatures whose validation is requested, as well as their relative position to the signed document(s).

Rows in the table show different XML and JSON elements, which in the validation request message may appear for incorporating signature(s), signed documents, or representations of the signed documents.

Each cell in the table indicates the required cardinality of the XML element and JSON element (or the number of items within the array if the JSON element is an array) shown in the header of the corresponding row, for the type of signature, located in a relative position to the signed document(s) as indicated in the header of the corresponding column. Cardinalities are indicated as in Table 3.

Table 4: Components for signatures and documents in validation requests messages for XML and JSON protocols based on OASIS specifications

Type of signature and relative position / **XML elements** / **JSON elements**	**CAdES attached**	**CAdES detached**	**Non-embedded XAdES**	**Embedded XAdES**	**PAdES**
`dss2:SignatureObject` `sigObject`	1	1	1	0..1 (see note)	0..1 (see note)

`dss2:Document`
`Items in doc array`
	0	0..1	*	1..*	0..1

`dss2:DocumentHash`
`Items in docHash array`
	0	0..1	*	*	0..1

`dss2:TransformedData`
`Items in transformed array`
	0	0	*	*	0

NOTE: This component is optional because even if the signature is embedded one could submit the hash of the document, and consequently the signature would be transferred in this component.					

5.1.4 Optional components

5.1.4.1 Container for optional components

5.1.4.1.1 Component semantics

The validation request message may also contain optional components. These components may be used for requesting validation of any type of AdES signature.

Below follows the list of the new components defined by the present document:

1) One component for identifying the signatures that the server is requested to validate in addition to the signature present or referenced within the signature object container. Clause 5.1.4.2.1.2 specifies semantic requirements for this component.

NOTE: Each signed document submitted to the server can be signed by more than one signature. This component allows the client to request to the server the validation of some of them.

2) Component for requesting validation of the signature(s) against a certain signature policy. Clause 5.1.4.2.2.1 specifies semantic requirements for this component.

3) Component for requesting detailed validation report(s) of the validation of the signature(s). Clause 5.1.4.2.3.1 specifies semantic requirements for this component. This component shall also allow indicating whether the validation report has to be signed by the server or not.

4) One component for passing to the server one or more time values, each one being, according to client's claim, a Proof of Existence of one signature present within the request. Clause 5.1.4.2.4.1 specifies semantic requirements for this component.

Below follows the list of the components already specified in DSS-X core v2.0 [1] that the present document re-uses:

1) One component for identifying one or more service policies under which the validation shall be conducted. Clause 5.1.4.3.1.1 specifies semantic requirements for this component.

2) One component for requesting the server to generate notifications using a certain language. Clause 5.1.4.3.2.1 specifies semantic requirements for this component.

3) One component for claiming the client's identity. Clause 5.1.4.3.3.1 specifies semantic requirements for this component.

4) One component for passing to the server XML schemas that it can need during the validation process. Clause 5.1.4.3.4.1 specifies semantic requirements for this component.

5) One component for requesting to set the validation time to a certain instant different from the current time. Clause 5.1.4.3.5.1 specifies semantic requirements for this component.

6) One component for requesting the server to return information on the validation time. Clause 5.1.4.3.6.1 specifies semantic requirements for this component.

7) One component for passing to the server validation material in case this is not present within the signature(s) to be validated. Clause 5.1.4.3.7.1 specifies semantic requirements for this component.

8) One component for requesting the server to return information on the signing time(s). Clause 5.1.4.3.8.1 specifies semantic requirements for this component.

9) Component for requesting the server to return the identity of the signer(s). Clause 5.1.4.3.9.1 specifies semantic requirements for this component.

10) In the case of requesting validation of XAdES signatures, one component requesting to the server to return the result of applying the set of transformations indicated within a certain `ds:Reference` element to the input document that such `ds:Reference` element references. Clause 5.1.4.3.10.1 specifies semantic requirements for this component.

11) In the case of requesting validation of XAdES signatures, request to return the results of the validation of any signed `ds:Manifest` present in these signatures. Clause 5.1.4.3.11.1 specifies semantic requirements for this component.

5.1.4.1.2 XML component

The `dss2:OptionalInputs` child element of `dss2:VerifyRequest` shall be an instance of `dss2:OptionalInputsVerifyType` redefined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1. The redefinition of `dss2:OptionalInputsVerifyType` has been copied in clause 4.3.2 for information.

The following children elements of `dss2:OptionalInputsVerifyType` type shall not be present:

1) The `dss2:AddTimeStamp` element.

2) The `dss2:ReturnAugmentedSignature` element.

3) The `dss2:ReturnTimeStampedSignature` element.

4) The `dss2:ReturnProcessingDetails` element.

5) The `dsb:Other` element.

NOTE 1: The three first elements in the former list are not allowed because this protocol supports validation, but not augmentation. The `dss2:ReturnProcessingDetails` element is not allowed because clients desiring to get detailed information on the validation process can request signed or unsigned validation report(s) using the `etsival:ReturnValidationReport` element. The `dsb:Other` element is not allowed for keeping a low degree of optionality.

In addition to that, the `etsival:ReturnAugmentedSignature` and `etsival:TSTokensQualityLevel` elements shall not be present either.

NOTE 2: The optional child element `etsival:ReturnAugmentedSignature` is the component for requesting augmenting of the signature. The protocol specified in this clause is the "validation protocol". The `etsival:TSTokensQualityLevel` requests to the server that it uses time-stamp tokens of a certain quality level during the augmentation process, if it requires them. They will certainly be used within the "augmentation protocol" and the "validation and augmentation protocol". These elements are present in the redefinition of the `dss2:OptionalInputsVerifyType` type for re-using this XML schema definition in the "validation and augmentation" protocol.

5.1.4.1.3 JSON component

The `optInp` child element of `VerifyReq` shall be an instance of `OptionalInputsVerifyType` defined as in JSON Schema file "19442jsonSchema.json", whose location is detailed in clause A.2, and is copied below for information.

```
  "OptionalInputsVerifyType": {
    "type": "object",
    "allOf": [

      {"$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-OptionalOutputsVerifyType"}

    ],

    "properties": {
      "processSigs": {
        "$ref": "#/definitions/SigsRefsType"
      },
      "useSigValPol":{
        "$ref":"#/definitions/SigValPolicyType "
      },
      "returnValReport": {
        "$ref": "#/definitions/ReturnValReportType"
      },
      "proofsOfExist": {
        "$ref": "#/definitions/ProofsOfExistenceType"
      },
      "etsiReturnAugmentedSig": {
        "level": "string",
        "format" : "uri"
      },
      "tstkQualityLevel": {
        "type": "string",
        "format": "uri"
      }
    }
  }
```

The following properties of `dss2-OptionalInputsVerifyType` type shall not be present:

1) The `addTimeStamp` element.

2) The `returnAugmented` element.

3) The `returnTimestamped` element.

4) The `returProcDetails` element.

5) The `dsb:Other` element.

In addition to that, the `etsiReturnAugmentedSig` and `tstkQualityLevel` elements shall not be present either.

NOTE: See notes 1 and 2 in clause 5.1.4.1.2 of the present document for justifications of these absences.

5.1.4.2 New components defined in the present document

5.1.4.2.1 Component for identifying the signatures to be processed (signatures-to-process-refs container)

5.1.4.2.1.1 Introduction

A validation request message may contain more than one signature; however, certain business processes may not require all of them to be processed. Consequently, requests messages need a mechanism for allowing the client to enumerate to the server the signatures that the client requests to process (validate in the case of this validation protocol, augment in the case of the augmentation protocol, or validate and augment in the validation and augmentation protocol). This mechanism is provided by the optional input that is fully specified in clause.

5.1.4.2.1.2 Component semantics

The signatures-to-process-refs component shall contain one reference per signature that the server is requested to process.

The signatures-to-process-refs component shall allow referencing any signature present in the validation request.

There are are three different ways for building a reference to one signature, namely:

1) Using the digest value of the signature computed using a specific digest algorithm. This mechanism may be used for referencing any type of signature. In this case, this component:

   - shall include one or more digest values (each one computed on one digital signature);

   - shall include the identification of one digest algorithm: the algorithm used for computing all the digest values; and

   - shall include the identifier of a canonicalization algorithm, if some of the signatures to validate are XAdES signatures.

2) Using a pointer to the embedding document and a XPath expression pointing to that specific embedded XAdES signature within the embedding document. This mechanism may only be used if the signature is an embedded XAdES signature.

3) Using the name of the field where the PAdES signature is embedded within the PDF document. This mechanism may only be used if the signature is a PAdES signature.

This component may contain references based on different referencing mechanisms if they are references to different signatures.

This component shall contain only one reference for one signature.

5.1.4.2.1.3 XML component

The `ProcessSignatures` optional input shall be an instance of `SignatureIdentifiersType` defined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1, and is copied below for information.

```
<!—targetNamespace=http://uri.etsi.org/19442/v1.1.1# -->

    <xs:element name="ProcessSignatures" type="etsival:SignaturesReferencesType"/>

    <xs:complexType name="SignaturesReferencesType">
        <xs:sequence>
            <xs:element ref="etsival:DigestReferences" minOccurs="0"/>
            <xs:element ref="etsivr:XAdESSignaturePtr" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="PAdESFieldName" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="DigestReferences" type="etsival:DigestReferencesType"/>

    <xs:complexType name="DigestReferencesType">
        <xs:sequence>
            <xs:element name="CanonicalizationMethod" type="xs:anyURI" minOccurs="0"/>
            <xs:element name="DigestMethod" type="xs:anyURI"/>
            <xs:element name="DigestValue" type="xs:base64Binary" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

This XML Schema piece references `etsivr:XAdESSignaturePtr` element, which is defined in ETSI TS 119 102-2 [11]. For the sake of completeness of the present document, below follows its definition, copied from the XML Schema of ETSI TS 119 102-2 [11].

```
<!—targetNamespace=http://uri.etsi.org/19102/v1.2.1# -->

    <xs:element name="XAdESSignaturePtr" type="etsivr:XAdESSignaturePtrType"/>

    <xs:complexType name="XAdESSignaturePtrType">
        <xs:sequence>
            <xs:element name="NsPrefixMapping" type="etsivr:NsPrefixMappingType" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="WhichDocument" type="xs:IDREF" use="optional"/>
        <xs:attribute name="XPath" type="xs:string" use="optional"/>
        <xs:attribute name="SchemaRefs" type="xs:IDREFS" use="optional"/>
    </xs:complexType>
    <xs:complexType name="NsPrefixMappingType">
        <xs:sequence>
            <xs:element name="NamespaceURI" type="xs:anyURI"/>
            <xs:element name="NamespacePrefix" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
```

The `DigestReferences` child element may be used for referencing any type of signature within the request.

The value of the `CanonicalizationMethod` child element shall be an URI identifying a canonicalization algorithm.

The value of the `DigestMethod` child element shall be an URI identifying a digest algorithm.

The value of the `DigestValue` child element shall be the base-64 encoded value of the digest of the referenced digital signature computed using the digest algorithm identified in `DigestMethod` child element's value.

The actual computation of the digest value shall be dependent on the type of signature and shall be performed as follows:

1) In case of CAdES signatures, the input to the digest value computation shall be one of the DER-encoded instances of `SignedInfo` type present within the CMS structure.

NOTE: A CMS structure can enclose several parallel CAdES signatures (each item in the `signerInfos` array, which is an instance of `SignedInfo` type, contains the digital signature value generated by a different private key).

2) In case of XAdES signatures, the input of the digest value computation shall be the result of applying the canonicalization algorithm identified within the `CanonicalizationMethod` child element's value to the corresponding `ds:Signature` element and its contents. The canonicalization shall be computed keeping this `ds:Signature` element as a descendant of the XML root element, without detaching it.

3) In case of PAdES signatures, the input of the digest value computation shall be the result of decoding the hexadecimal string present within the `Contents` field of the `Signature` PDF dictionary enclosing one PAdES digital signature.

`XAdESSignaturePtr` child element shall only be used for referencing XAdES signatures.

`XAdESSignaturePtr` child element is an instance of `XAdESSignaturePtrType` type, whose requirements are the same as the requirements for `dss2:SignaturePtrType` specified in clause 4.2.9.2 of DSS-X core v2.0 [1], with the following exceptions:

1) The attribute `WhichDocument` may be absent. If this attribute is present, its value shall be as specified in clause 4.2.9 of DSS-X core v2.0 [1] and the `XAdESSignaturePtr` element shall reference a signature embedded in some input document. If this attribute is absent, the `XAdESSignaturePtr` element shall reference one of the signatures placed within the signature object container.

2) The `SchemaRefs` attribute shall have the same semantics and usage as `SchemaRefs` attribute in `dss2:DocumentBaseType` specified in clause 4.2.2.2 of DSS-X core v2.0 [1].

`PAdESFieldName` child element shall only be used for referencing PAdES signatures.

The value of `PAdESFieldName` child element shall be the name of the PDF field where the referenced PAdES signature is present within the PDF signed document.

5.1.4.2.1.4 JSON component

The `processSigs` element shall be an instance of `SigsRefsType` defined as in JSON Schema file "19442jsonSchema.json", whose location is detailed in clause A.2, and is copied below for information.

```
"NsPrefixMayppingType": {
    "type": "object",
    "properties": {
        "NsURI": {
            "type": "string",
            "format": "uri"
        },
        "NsPrefix": {"type": "string"}
    },
    "required": ["NsURI"]
},

"XAdESSignaturePtrType": {
    "type": "object",
    "properties": {
        "nsPrefixMapping": {
            "type": "array",
            "items": {
                "$ref": "#/definitions/NsPrefixMayppingType"
            }
        },
        "whichDoc": {"type": "string"},
        "xpath": {"type": "string"},
        "schemaRefs": {"type": "string"}
    }
}

"SigsRefsType": {
    "type": "object",
    "properties": {
        "digRefs": {
            "type": "object",
            "properties": {
                "digVals": {
                    "type": "array",
                    "items": {
                    "type": "string"
                    }
                },
                "digAlg": {
                    "type": "string"
                },
                "canAlg": {
                    "type": "string"
                }
            }
            "required": ["digVals", "digAlg"]
        },
        "padesFieldNames": {
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "xadesSigPtrs": {
            "type": "array",
            "items": {
                "$ref": "#definitions/XAdESSignaturePtrType"
            }
        }
    }
}
```

Below follow the requirements that apply to this element:

1) Each item within `digVals` array shall have the same requirements as the `DigestValue` element in the previous clause, including those ones that apply to the computation of their values.

2) The property `digAlg` shall have the same requirements as the `DigestMethod` element in the previous clause.

3) The property `canAlg` shall have the same requirements as the `CanonicalizationMethod` element in the previous clause.

The `xadesSigPtrs` element may only be used for referencing XAdES signatures.

The `xadesSigPtrs` element is an array of objects instance of `dss2-SignaturePtrType` specified in clause 4.2.9.1 of DSS-X core v2.0 [1], with the following exception:

1) The element `whichDoc` may be absent. If this element is present, its value shall be as specified in clause 4.2.9.1 of DSS-X core v2.0 [1] and the corresponding item of the array shall reference a signature embedded in some input document. If this element is absent, the corresponding item of the array shall reference one of the signatures placed within the signature object container.

The `pAdESFieldNames` array may only be used for referencing PAdES signatures.

The value of each item within the `pAdESFieldNames` array shall be the name of the PDF field where the referenced PAdES signature is present within the PDF signed document.

5.1.4.2.2 Component for requesting validation against a certain signature policy

5.1.4.2.2.1 Component semantics

This component shall provide means for unambiguously identifying the signature validation policy against which the client requests to validate the digital signature(s).

This component shall also allow the client to indicate locations where the signature validation policy can be downloaded from, in case the client wants to indicate them to the server.

5.1.4.2.2.2 XML component

The element that shall contain the signature validation policy to be used shall be the `UseSignatureValidationPolicy` element.

The `UseSignatureValidationPolicy` element shall be defined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1, and is copied below for information.

```
<!—targetNamespace="http://uri.etsi.org/19442/v1.1.1#" -->

  <xs:element name="UseSignatureValidationPolicy" type="etsival:UseSignatureValidationPolicyType"/>

  <xs:complexType name="UseSignatureValidationPolicyType">
    <xs:sequence>
      <xs:element name="SignatureValidationPolicyID" type="xs:anyURI"/>
      <xs:element name="SignaturePolicyLocation" type="xs:anyURI" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
```

The `SignatureValidationPolicyID` child element shall have as value the unique identifier of the signature validation policy as an URI. If the identifier of the signature validation policy is an OID, then the value of this element shall be an URN indicating the value of the aforementioned OID as specified in IETF RFC 3061 [14].

Every `SignatureValidationPolicyLocation` child element shall have as value one location where the signature validation policy document can be accessed, as an URI value.

5.1.4.2.2.3 JSON component

The element that shall request the server to validate the signature(s) against a certain signature validation policy shall be the `useSigValPol` element.

The `useSigValPol` element shall be an instance of `SigValPolicyType` defined as in JSON Schema file "19442jsonSchema.json, whose location is detailed in clause A.2, and is copied below for information.

```
"SigValPolicyType": {
  "type": "object",
  "properties": {
    "sigValPolID": {
      "type": "string"
    },
    "sigValPolLocs": {
      "type": "array",
      "items": {
        "type": "string"
      }
    }
  }
}
```

The `SigValPolID` child element shall have as value the unique identifier of the signature validation policy as an URI. If the identifier of the signature validation policy is an OID, then the value of this element shall be an URN indicating the value of the aforementioned OID as specified in IETF RFC 3061 [14].

Every `SigValPolLos` child element shall have as value one location where the signature validation policy document can be accessed, as an URI value.

5.1.4.2.3 Component for requesting a detailed validation report (signed or unsigned)

5.1.4.2.3.1 Component semantics

This component shall provide means for requesting to the server the generation and return of a detailed validation report for each validated signature.

This component shall allow the client to identify that the validation report is conformant to a certain specification.

The identifier for requesting the generation of a validation report as specified in ETSI TS 119 102-2 [11], shall be: `http://uri.etsi.org/19442/1910202v010201`.

This component shall allow requesting to the server signing the aforementioned validation report.

5.1.4.3.2 XML component

The element that shall request to the server returning one detailed validation report for each signature validated shall be the `ReturnValidationReport` element.

The `ReturnValidationReport` element shall be defined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1, and is copied below for information.

```
<!—targetNamespace="http://uri.etsi.org/19442/v1.1.1#" →

    <xs:element name="ReturnValidationReport" type="etsival:ReturnValidationReportType" />

    <xs:complexType name="ReturnValidationReportType">
        <xs:sequence>
            <xs:element name="AsSpecifiedBy" type="xs:anyURI"/>
        </xs:sequence>
        <xs:attribute name="SignIt" type="xs:boolean" default="false"/>
    </xs:complexType>
```

The value of the `AsSpecifiedBy` element shall be an URI identifying the validation report that the client is requesting.

A "`true`" value of the attribute `SignIt` shall indicate that the client is requesting that the server signs the validation report. A "`false`" value or absence of this attribute shall indicate that the client is not requesting that the server signs the validation report.

5.1.4.3.3 JSON component

The `returnValReport` element shall be an instance of `ReturnValReportType` defined as in JSON Schema file "19442jsonSchema.json", whose location is detailed in clause A.2, and is copied below for information.

```
  "ReturnValReportType": {
    "type": "object",
    "properties": {
      "asSpecifiedBy":{
        "type": "string",
        "format": "uri"
      },
      "signIt":{"type": "boolean"}
  }
```

The value of the `asSpecifiedBy` element shall be an URI identifying the validation report that the client is requesting.

A "`true`" value of the component `signIt` shall indicate that the client is requesting that the server signs the validation report. A "`false`" value or absence of this component shall indicate that the client is not requesting that the server signs the validation report.

5.1.4.2.4 Component for passing proofs of existence of one or more signatures

5.1.4.2.4.1 Component semantics

This component shall have one or more tuples. Each tuple shall contain two components, namely:

1) One component whose value is a time instant value. This time instant shall be considered by the server as a proof of existence of the signature referenced in the other component of the tuple.

2) One component referencing one signature.

5.1.4.2.4.2 XML component

The `ProofsOfExistence` optional input shall be an instance of `ProofsOfExistenceType` defined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1, and is copied below for information.

```
<!—targetNamespace="http://uri.etsi.org/19442/v1.1.1#" -->

    <xs:element name="ProofsOfExistence" type="etsival:ProofsOfExistenceType"/>
    <xs:complexType name="ProofsOfExistenceType">
        <xs:sequence>
            <xs:element ref="etsival:ProofOfExistence" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:element name="ProofOfExistence" type="etsival:ProofOfExistenceType"/>
    <xs:complexType name="ProofOfExistenceType">
        <xs:sequence>
            <xs:element name="Time" type="xs:dateTime"/>
            <xs:element name="SignatureReference" type="etsivr:SignatureReferenceType"/>
        </xs:sequence>
    </xs:complexType>
```

This XML Schema piece references `etsivr:SignatureReferenceType` type, which is defined in ETSI TS 119 102-2 [11]. For the sake of completeness of the present document, below follows its definition, copied from the XML Schema of ETSI TS 119 102-2 [11].

```
<!—targetNamespace=http://uri.etsi.org/19102/v1.2.1# -->

    <xs:complexType name="SignatureReferenceType">
        <xs:choice>
            <xs:sequence>
                <xs:element name="CanonicalizationMethod" type="xs:anyURI" minOccurs="0"/>
                <xs:element name="DigestMethod" type="xs:anyURI"/>
                <xs:element name="DigestValue" type="xs:base64Binary"/>
            </xs:sequence>
            <xs:element ref="etsivr:XAdESSignaturePtr"/>
            <xs:element name="PAdESFieldName" type="xs:string"/>
            <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:choice>
    </xs:complexType>
```

`ProofsOfExistence` element shall be a sequence of `ProofOfExistence` children elements.

`SignatureReference` child element of `ProofOfExistence` shall be a reference to the signature whose Proof of Existence the client sends to the server. Any of the three mechanisms specified in clause 5.1.4.2.1.3 may be used for referencing the signature, depending on the type of the referenced signature.

`Time` child element of `ProofOfExistence` shall indicate the time and date when the client claims the referenced signature existed (Proof of Existence).

5.1.4.2.4.3 JSON component

The `proofsOfExist` element shall be an instance of `ProofsOfExistenceType` defined as in JSON Schema file "19442jsonSchema.json", whose location is detailed in clause A.2, and is copied below for information

```
  "ProofsOfExistenceType": {
    "type": "array",
    "items": {
      "type": "object",
      "properties": {
        "time": {
          "type": "string"
        },
        "sigRef": {
          "type": "object",
          "$ref": "#/definitions/SignatureReferenceType"
        }
      }
    }
  }

  "SignatureReferenceType": {
    "type": "object",
    "properties": {
      "digRef": {
        "type": "object",
        "properties": {
          "digVal": {
            "type": "string",
            "contentEncoding": "base64"
          },
          "digAlg": {
            "type": "string",
            "format": "uri"
          },
          "canAlg": {
            "type": "string",
            "format": "uri"
          }
        }
        "required": ["digVal", "digAlg"]
      },
      "padesFieldName": {
        "type": "string"
      },
      "xadesSigPtr": {
        "$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-SignaturePtrType"
      }
    }
  }
```

Instances of `ProofsOfExistenceType` type shall be an array of items. Each item shall be a tuple of two components.

`sigRef` component of the tuple shall be a reference to the signature whose Proof of Existence the client sends to the server. Any of the three mechanisms specified in clause 5.1.4.2.1.4 may be used for referencing the signature, depending of the type of the referenced signature.

`time` component of the tuple shall indicate the time and date when the client claims the referenced signature existed (Proof of Existence). Its value shall be a string representing an UTC Time as represented in instances of `xs:dateTime` type in XML schema.

The client shall be fully responsible of submitting accurate Proofs of Existence to the server.

5.1.4.3 Components re-used from DSS-X core v2.0

5.1.4.3.1 Component for identifying under which service policy the validation has to be conducted

5.1.4.3.1.1 Component semantics

This component shall contain a non-ambiguous identifier of the service policy under which the client requests the server to validate the signature.

5.1.4.3.1.2 XML component

The element that shall identify under which service policy the validation has to be conducted shall be the `dsb:ServicePolicy` element specified in clause 4.1.8.2 of DSS-X core v2.0 [1].

NOTE: The service policy is not the same as signature policy. The server defines the service policy, and one server can have different service policies offering different features to its clients.

5.1.4.3.1.3 JSON component

The element that shall identify under which service policy the validation has to be conducted shall be the `policy` element. The contents of this element shall be as specified in clause 4.1.8.1 of DSS-X core v2.0 [1].

5.1.4.3.2 Component for requesting notifications in a certain language

5.1.4.3.2.1 Component semantics

This component shall identify a language by a string-valued identifier, whose value shall be one of the identifiers built and registered as specified in IETF RFC 5646 [12].

5.1.4.3.2.2 XML component

The element that shall request the server to return notifications in a certain language shall be the `dsb:Language` element specified in clause 4.1.8.2 of DSS-X core v2.0 [1].

The value of this element shall be string compliant with the values defined in IETF RFC 5646 [12].

5.1.4.3.2.3 JSON component

The element that shall request the server to return notifications in a certain language shall be the `lang` element. The contents of this element shall be as specified in clause 4.1.8.1 of DSS-X core v2.0 [1].

The value of this element shall be string compliant with the values defined in IETF RFC 5646 [12].

5.1.4.3.3 Component for allowing the client to claim its identity

5.1.4.3.3.1 Component semantics

This component shall provide the identity of the client as a string-valued name.

This component may include sub-components for supporting names federation.

This component may include one sub-component for identifying the format of the string-valued name representing the identity of the client.

This component may include one sub-component for integrating to the string-valued name representing the identity of the client, a different name identifier that has been established by the validation service itself for the client.

This component may also include one sub-component for incorporating any type of additional supporting information for the string-valued name representing the identity of the client.

5.1.4.3.3.2 XML component

The element that shall allow the client claiming for an identity shall be the `dss2:ClaimedIdentity` element specified in clause 4.3.9.2 of DSS-X core v2.0 [1].

NOTE: The `dss2:ClaimedIdentity` element builds on `saml2:NameID` element and it incorporates all the XML descendant elements and attributes for matching the semantic requirements in clause 5.1.4.3.3.

5.1.4.3.3.3 JSON component

The element that shall allow the client claiming for an identity shall be `claimedIdentity` , an instance of `dss2-ClaimedIdentityType`. This type is specified in clause 4.3.9.1 of DSS-X core v2.0 [1].

5.1.4.3.4 Component for passing schemas

5.1.4.3.4.1 Component semantics

This component shall contain one or more documents, each one containing a schema.

5.1.4.3.4.2 XML component

The element that shall request the server to return the identity of the signer shall be the `dss2:Schemas` element as specified in clause 4.3.10.2 of DSS-X core v2.0 [1].

5.1.4.3.4.3 JSON component

The element that shall request the server to return the identity of the signer shall be `schemas` element, of type `dss2-SchemasType`, which is specified in clause 4.3.10.1 of DSS-X core v2.0 [1].

5.1.4.3.5 Component for requesting to set the validation time to a certain value

5.1.4.3.5.1 Component semantics

This component shall provide means for indicating to the server that the validation time is either the current time (the time when the server performs the signature validation) or a certain time in the past.

5.1.4.3.5.2 XML component

The element that shall allow to set the validation time to a certain instant different from current time shall be the `dss2:UseVerificationTime` element instance of `dss2:UseVerificationTimeType` specified in clause 4.3.25.2 of DSS-X core v2.0 [1].

The value of `dss2:SpecificTime` child element shall be expressed as Coordinated Universal Time (UTC): its value shall contain year with four digits, month, day, hour, minute, second (without decimal fraction) and the UTC designator "Z". The time scale shall be based on the second.

5.1.4.3.5.3 JSON component

The element that shall allow to set the validation time to a certain instant different from current time shall be the `useVerificationTime` element, instance of `dss2-UseVerificationTimeType`. This type is specified in clause 4.3.25.1 of DSS-X core v2.0 [1].

The value of `specTime` child element shall be expressed as Coordinated Universal Time (UTC): its value shall contain year with four digits, month, day, hour, minute, second (without decimal fraction) and the UTC designator "Z". The time scale shall be based on the second

5.1.4.3.6 Component for requesting to return the validation time

5.1.4.3.6.1 Component semantics

This component shall provide means for requesting to the server to return within the response an indication of the validation time.

5.1.4.3.6.2 XML component

The element that shall allow to request to the server to return the validation time, shall be the `dss2:ReturnVerificationTimeInfo` element specified in clause 4.3.5.2 of DSS-X core v2.0 [1].

5.1.4.3.6.3 JSON component

The element that shall allow to request to the server to return the validation time, shall be the `returnVerificationTime` element. The contents of this element shall be as specified in clause 4.3.5.1 of DSS-X core v2.0 [1].

5.1.4.3.7 Component for passing validation material to the server

5.1.4.3.7.1 Component semantics

This element shall convey any type of validation material that the client decides to pass to the server. It shall provide mechanisms for passing to the server X509 certificates, Attribute certificates, CRLs, OCSP responses, or other type of validation data.

5.1.4.3.7.2 XML component

The element that shall allow the client to pass validation material to the server shall be the `dss2:AdditionalKeyInfo` element instance of `dss2:AdditionalKeyInfoType` as specified in clause 4.3.28.2 of DSS-X core v2.0 [1].

5.1.4.3.7.3 JSON component

The element that shall allow the client to pass validation material to the server shall be the `addKeyInfo` element, instance of `dss2-AdditionalKeyInfoType`. This type is specified in clause 4.3.28.1 of DSS-X core v2.0 [1].

5.1.4.3.8 Component for requesting return of the signing time

5.1.4.3.8.1 Component semantics

This component shall provide means for requesting to the server to return within the response an indication of the signing time for each validated signature.

5.1.4.3.8.2 XML component

The element that shall allow to request to the server to return an indication of the signing time shall be the `dss2:ReturnSigningTimeInfo` element specified in clause 4.3.5.2 of DSS-X core v2.0 [1].

5.1.4.3.8.3 JSON component

The element that shall allow to request to the server to return an indication of the signing time shall be the `returnSigningTime` element. The contents of this element shall be as specified in clause 4.3.5.1 of DSS-X core v2.0 [1].

5.1.4.3.9 Component for requesting the server to return the identity of the signer

5.1.4.3.9.1 Component semantics

This element shall convey an indication to the server for returning the identity of the signer(s) for each validated signature.

5.1.4.3.9.2 XML component

The element that shall request the server to return the identity of the signer shall be the `dss2:ReturnSignerIdentity` element specified in clause 4.3.5.2 of DSS-X core v2.0 [1].

5.1.4.3.9.3 JSON component

The element that shall request the server to return the identity of the signer shall be the `returnSigner` element. The contents of this element shall be as specified in clause 4.3.5.1 of DSS-X core v2.0 [1].

5.1.4.3.10 Component for requesting the server to return the result of transforming the input document

5.1.4.3.10.1 Component semantics

This component shall not be present when requesting validation of CAdES or PAdES signatures.

This component shall request to the server to return the result of applying the set of transformations indicated within a certain `ds:Reference` element of a XAdES signature to the input document that such `ds:Reference` element references.

This component shall provide means for identifying one or more signed documents whose transformed versions the client requests that the server incorporates into the validation response.

5.1.4.3.10.2 XML component

The element that shall request the server to return the result of transforming one input document shall be the `dss2-ReturnTransformedDocument` element instance of `dss2:ReturnTransformedDocumentType` specified in clause 4.3.33.2 of DSS-X core v2.0 [1].

5.1.4.3.10.3 JSON component

The element that shall request the server to return the result of transforming one input document shall be the `returnTransformed` element, instance of `dss2-ReturnTransformedDocumentType`. This type is specified in clause 4.3.33.1 of DSS-X core v2.0 [1].

5.1.4.3.11 Component for requesting to return the validation of signed ds:Manifest in XAdES signatures

5.1.4.3.11.1 Component semantics

This component shall not be present when requesting validation of CAdES or PAdES signatures.

This component shall notify that the client requests that the server verifies all the `ds:Manifest` elements present within the al the validated XAdES signatures.

The server shall interpret the absence of this component or its presence set to value `"false"` as absence of this request.

5.1.4.3.11.2 XML component

The element that shall request the server to return the result of transforming one input document shall be the `dss2:VerifyManifests` element specified in clause 4.3.5.2 of DSS-X core v2.0 [1].

5.1.4.3.11.3 JSON component

The element that shall request the server to return the result of transforming one input document shall be `verifyManifests`, set to value `"true"`.

5.2 Response message

5.2.1 Component for responding to a validation request

5.2.1.1 Component semantics

The validation response message resulting from one request of validation of AdES signature(s), shall contain one component for notifying the global validation result.

The validation response message may contain one or more signature results containers. Clause 5.2.3.1 specifies this component and contains references to clauses that specify requirements for its sub-components.

NOTE: The response message to a minimum signature validation request with only one signature and without any optional input, unless otherwise stated by the service policy applied, is a response message with a global result component and the component identifying the response as compliant with the "validation" protocol defined in the present document.

The validation response message shall contain one or more components identifying protocols and/or profiles that the response message is compliant with. The first one of such components shall have the following URI as value, identifying the response as one that has been built using the "validation" protocol specified in the present document:

- `http://uri.etsi.org/19442/v1.1.1/validationprotocol#`.

This message may contain a component whose value is an identifier of the message itself.

This message may contain component whose value is the identifier of the request message that this message is the response of.

5.2.1.2 XML component

The element that shall be the component for responding to the validation request of AdES signature(s) shall be the root element of the message `dss2:VerifyResponse` as specified in clause 4.2.11.2 of DSS-X core v2.0 [1], where the `optOutp` container of optional outputs may also include the additional components specified in clause 5.2.3.1.3 of the present document.

The `dsb:AppliedProfile` array shall have one or more items. The first one shall have the value `http://uri.etsi.org/19442/v1.1.1/validationprotocol#`, identifying the response as a validation response compliant with the validation protocol specified in the present document.

5.2.1.3 JSON component

The element that shall implement the verify response message for the JSON binding of the protocol shall be the element `VerifyResp` element. This element shall be an instance of the `VerifyResponseType`, defined as in JSON Schema file "19442jsonSchema.json", whose location is detailed in clause A.1, and is copied below for information.

```
"VerifyResponseType": {
  "type": "object",
  "properties": {
    "result": {
      "$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dsb-ResultType"
    },
    "profile": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "reqID": {
      "type": "string"
    },
    "respID": {
      "type": "string"
    },
    "optOutp": {
      "$ref": "#/definitions/OptionalOutputsVerifyType"
    }
  }
}
```

The profile, `reqID`, and `respID` elements shall be as specified in clause 4.1.11.1 of DSS-X core v2.0 [1].

The `profile` array shall have one or more items as specified in clause 4.2.10.1 of DSS-X core v2.0 [1]. The first one shall have the value `http://uri.etsi.org/19442/v1.1.1/validationprotocol#`, identifying the response as a validation response compliant with the validation protocol specified in the present document.

The `optOut` shall be as specified in clause 5.2.3.1.3 of the present document.

The `result` element shall be as specified in clauses 5.2.2.1 and 5.2.2.3 of the present document.

5.2.2 Component for the global validation result

5.2.2.1 Component semantics

This component shall contain a major result, which shall report whether the server has been able to perform its task, regardless the results obtained. This component may also contain a minor result providing additional information on the task performed by the server.

For details of the values of its result major and result minor, see clause 8.4.3.1 of the present document.

5.2.2.2 XML component

The element that within the response shall notify the validation result shall be the `dsb:Result` element specified in clause 4.1.7.2 of DSS-X core v2.0 [1].

5.2.2.3 JSON component

The element that within the response shall notify the validation result shall be an instance of the `dsb-ResultType` type specified in clause 4.1.7.2 of DSS-X core v2.0 [1].

5.2.3 Optional components

5.2.3.1 Container for optional components

5.2.3.1.1 Component semantics

The validation response message may also include one or more signature results containers.

Each signature result container shall include one or more optional outputs resulting from the validation of the corresponding signature.

Below follows the list of the new components defined by the present document:

1) The processing signature results container.

2) One component for referencing the signature that the rest of optional outputs within the signature results container correspond to.

3) One component for notifying the signature policy applied during the validation. Clause 5.2.3.2.3 specifies requirements for this component.

4) One component for notifying the set of signature policies supported by the server. Cause 5.2.3.2.4 specifies requirements for this component.

5) One component for returning the signed or unsigned detailed validation report. Clause 5.2.3.2.5 specifies requirements for this component.

Below follows the list of the components already specified in DSS-X core v2.0 [1] that the present document re-uses:

1) One component for identifying the service policy under which the validation was conducted. Clause 5.2.3.3.1 specifies requirements for this component.

2) One component for returning the result of transforming the input document. Clause 5.2.3.3.5 specifies requirements for this component.

3) One component for returning the results of validating any signed `ds:Manifest` present in the signature(s). Clause 5.2.3.3.6 specifies requirements for this component.

4) One component for indicating the signing time. Clause 5.2.3.3.3 specifies requirements for this component.

5) One component for indicating the time when the validation was conducted (validation time). Clause 5.2.3.3.2 specifies requirements for this component.

6) One component for indicating the identity of the signer(s). Clause 5.2.3.3.4 specifies requirements for this component.

Some of the optional components affect all the signatures validated, namely:

- The component for notifying the signature policy applied during the validation.
- The component for notifying the set of signature policies supported by the server.
- The component for identifying the service policy under which the validations was conducted.
- The component for indicating the time when the validations were conducted (validation time).

The rest of the optional components affect one of the signatures validated, and may also be included within the component providing details that correspond to the validation of the validated signature.

5.2.3.1.2 XML component

The `dss2:OptionalOutputs` child element of `dss2:VerifyResponse` shall be an instance of `dss2:OptionalOutputsVerifyType` defined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1. The redefinition of `dss2:OptionalOutputsVerifyType` has been copied in clause 4.3.2 for information.

The following elements shall not appear within the response message of this protocol:

- `dss2:DocumentWithSignature` element.
- `dss2:AugmentedSignature` element.
- `dss2:Schemas` element.
- `dss2:ProcessingDetails` element.

NOTE: The optional children elements `dss2:DocumentWithSignature` and `AugmentedSignature` are containers for augmented signatures. The protocol specified in this clause is the "validation protocol", and consequently it does not manage augmented signatures. These elements will certainly be used within the "augmentation" protocol and the "validation and augmentation" protocol.

5.2.3.1.3 JSON component

The `optOutp` child element of `VerifyReq` shall be an instance of `OptionalOutputsVerifyType` defined as in JSON Schema file "19442jsonSchema.json", whose location is detailed in clause A.2, and is copied below for information.

```
"OptionalOutputsVerifyType": {
  "type":"object",
  "allOf": [
    {"$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-OptionalOutputsVerifyType"}
  ],
  {"properties":
    "validationReport": {
      "$ref": "#/definitions/ValReportContainerType"
    },
    "resForEachSignature": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/ResultsForOneSignatureType"
      }
    },
    "appliedSigValPolicy": {
      "type": "string",
      "format": "uri"
    },
    "availableSigValPols":{
      "type": "array",
      "items": {
        "type": "string",
        "format": "uri"
      }
    }
  }
}
```

The following elements shall not appear within the response message of this protocol:

- `docWithSignature` element.
- `augSig` element.
- `schemas` element.
- `procDetails` element.

NOTE: The optional children elements `docWithSignature` and `augSig` are containers for augmented signatures. The protocol specified in this clause is the "validation protocol", and consequently it does not manage augmented signatures. These elements will certainly be used within the "augmentation" protocol and the "validation and augmentation" protocol.

Each item of the `resultsForEachSignature` array shall be an instance of `ResultsForOneSignatureType` type as specified in clause 5.2.3.2.1.3 of the present document.

The `appliedSigValPolicy` is specified in clause 5.2.3.2.3.3 of the present document.

The `availableSigValPols` is specified in clause 5.2.3.2.4.3 of the present document.

5.2.3.2 New components defined in the present document

5.2.3.2.1 Signature processing results container

5.2.3.2.1.1 Component semantics

This component shall have one child summarizing the result of the validation of one signature. Its contents shall be as specified in clause 8.4.3.1 of the present document.

This component shall have one child for referencing the signature whose validation details it includes.

This component may have one child indicating the signer identity.

This component may have one child indicating the signing time of the validated signature.

This component may have one child may have one child containing one detailed validation report, which may be signed.

If the validated signature is a XAdES signature, this component may have one or more children each one providing information on the validation of `ds:Manifest` children.

If the validated signature is a XAdES signature, this component may have one or more children each one containing one transformed document.

5.2.3.2.1.2 XML component

The new `etsival:ResultsForOneSignature` element shall include elements providing details on the validation of one signature. It shall be an instance of `ResultsForOneSignatureType` defined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1, and is copied below for information.

```
<xs:element name="ResultsForOneSignature" type="etsival:ResultsForOneSignatureType"/>

<xs:complexType name="ResultsForOneSignatureType">
      <xs:sequence>
        <xs:element ref="dsb:Result"/>
        <xs:element name="SignatureReference" type="etsivr:SignatureReferenceType" minOccurs="0"/>
        <xs:element ref="dss2:SignerIdentity" minOccurs="0"/>
        <xs:element ref="dss2:SigningTimeInfo" minOccurs="0"/>
        <xs:element ref="etsival:ValidationReport" minOccurs="0"/>
        <xs:element ref="dss2:VerifyManifestResults" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="dss2:TransformedDocument" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="etsival:AugmentSignatureResult" minOccurs="0" />
      </xs:sequence>
</xs:complexType>
```

The `etsival:ResultsForOneSignature` element shall not be empty.

The `SignatureReference` child element shall be present when the response contains more than one `etsival:ResultsForOneSignature` elements.

The element `etsival:AugmentSignatureResult` shall not be present in this protocol.

NOTE: The `etsival:AugmentSignatureResult` element is present in the definition of `ResultsForOneSignatureType` because it is used in the "augmentation" and the "validation and augmentation" protocols.

5.2.3.2.1.3 JSON component

Each item of the `resForEachSignature` array shall include elements providing details on the validation of one signature. Each item shall be an instance of `ResultsForOneSignatureType` defined as in JSON Schema file "19442jsonSchema.json", whose location is detailed in clause A.2, and is copied below for information.

```
"ResultsForOneSignatureType":{
    "type": "object",
    "properties": {
        "result": {
            "$ref": "<DSSXBASESCHEMAFILELOCATION>#/definitions/dsb-ResultType"
        },
        "sigRef" : {
            "$ref": "#/definitions/SignatureReferenceType"
        },
        "signerIdentity" : {
            "$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/saml2rw-NameIDType"
        },
        "signingTimeInfo": {
            "$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-SigningTimeInfoType"
        },
        "validationReport": {
            "$ref": "#/definitions/ValReportContainerType"
        },
        "manifestValResults":{
            "$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-VerifyManifestResultsType"
        },
        "transformed":{
            "type": "array",
            "items": {"$ref:" "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-TransformedDocumentType"}
        },
    "augmentSigResult": {
            "$ref": "#definitions/AugmentSigResultType"
        },
    },
    "required": ["result"]
}
```

The `sigRef` component shall be present when the `resForEachSignature` array contains more than one items.

The `augmentSigResult` property shall not be present in this protocol.

NOTE: The `augmentSigResult` element is present in the definition of `ResultsForOneSignatureType` because it is used in the "augmentation" and the "validation and augmentation" protocols.

5.2.3.2.2 Component for referencing the validated signature

5.2.3.2.2.1 Component semantics

This component shall be able to identify one of the signatures present in the request message, and consequently shall contain one reference to one signature.

5.2.3.2.2.2 XML component

The `SignatureReference` component shall be an instance of `etsivr:SignatureReferenceType` defined in XML Schema file of ETSI TS 119 102-2 [11].

5.2.3.2.2.3 JSON component

The `sigRef` component shall be an instance of `SignatureReferenceType` defined as in JSON Schema file "19442jsonSchema.json", whose location is detailed in clause A.2, and has been copied in clause 5.1.4.2.4.3.

5.2.3.2.3 Component for notifying the signature policy applied during the validation

5.2.3.2.3.1 Component semantics

This component shall provide means for returning to the client the identifier of the signature validation policy applied by the server for validating the signature(s).

This component shall appear in the response to requests of validation of digital signatures that incorporate the component requesting the server to validate the signature(s) using a certain signature validation policy, specified in clause 5.1.4.2.2.1 of the present document, if the server has been able to check all the constraints defined in the signature validation policy identified in the present component, on all the validated signatures. It may also appear in responses to requests that do not incorporate the aforementioned component.

5.2.3.2.3.2 XML component

The element that shall notify to the client the signature validation policy against which the signature has been validated shall be the `AppliedSignatureValidationPolicy` element.

The `AppliedSignatureValidationPolicy` element shall be defined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1, and is copied below for information.

```
<!—targetNamespace="http://uri.etsi.org/19442/v1.1.1#" -->

<xs:element name="AppliedSignatureValidationPolicy" type="xs:anyURI"/>
```

The `AppliedSignatureValidationPolicy` element shall be a direct child of the `dss2:OptionalOutputs` optional child of `dss2:VerifyResponse`.

NOTE: When a `dss2:VerifyRequest` includes the `UseSignatureValidationPolicy` optional input, it is instructing the server to use the same signature validation policy for validating all the signatures whose validation is requested. The server applies the same signature validation policy for all the signatures whose validation is requested. Consequently one single instance of `AppliedSignatureValidationPolicy` is enough to notify its identifier, and this value is always placed out of all the `ResultsForOneSignature` children.

The `AppliedSignatureValidationPolicy` element shall have as value the unique identifier of the signature validation policy used by the server for validating the signature as an URI. If the identifier of the signature validation policy is an OID, then the value of this element shall be an URN indicating the value of the aforementioned OID as specified in IETF RFC 3061 [14].

5.2.3.2.3.3 JSON component

The element that shall notify to the client the signature validation policy against which the signature has been validated shall be the `appliedSigValPol`.

The value of this element shall be the unique identifier of the signature validation policy used by the server for validating the signature as an URI. If the identifier of the signature validation policy is an OID, then the value of this element shall be an URN indicating the value of the aforementioned OID as specified in IETF RFC 3061 [14].

5.2.3.2.4 Component for notifying the signature policies under which the server can conduct validation

5.2.3.2.4.1 Component semantics

This component shall provide means for returning to the client the identifiers of the signature validation policies under which the server can validate signatures.

This component shall appear in the response to requests of validation of digital signatures that incorporate the component requesting the server to validate the signature(s) using a certain signature validation policy, specified in clause 5.1.4.2.2.1 of the present document.

This component shall only appear if the server is not able to validate the signature under the signature validation policy requested by the client in the aforementioned component of the validation request.

5.2.3.2.4.2 XML component

The element that shall notify to the client the signature validation policies under which the server can conduct validation of digital signatures shall be the `AvailableSignatureValidationPolicies` element.

The `AvailableSignatureValidationPolicies` element shall be defined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1, and is copied below for information.

```
<!—targetNamespace="http://uri.etsi.org/19442/v1.1.1#" →

  <xs:element name="AvailableSignatureValidationPolicies"
type="AvailableSignatureValidationPoliciesType"/>
  <xs:complexType name="AvailableSignatureValidationPoliciesType">
    <xs:sequence>
      <xs:element name="AvailableSignatureValidationPolicyID" type="xs:anyURI" minOccurs="1"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
```

The `AvailableSignatureValidationPolicies` element shall be a direct child of the `dss2:OptionalOutputs` optional child of `dss2:VerifyResponse`.

NOTE: The information of the signature validation policies against which the server may validate signatures is independent of the validated signatures. Consequently one single instance of `AvailableSignatureValidationPolicies` is enough to notify its identifier, and this value is always placed outside of all the `ResultsForOneSignature` children.

Each `AvailableSignatureValidationPolicyID` child element shall have as value the unique identifier of one signature validation policy against which the server is able to validate digital signatures, as an URI. If the identifier of the signature validation policy is an OID, then the value of this element shall be an URN indicating the value of the aforementioned OID as specified in IETF RFC 3061 [14].

5.2.3.2.4.3 JSON component

In the JSON protocol derived from OASIS specifications the element that shall notify to the client the signature validation policies under which the server can conduct validation of digital signatures shall be the `availableSigValPols` element.

Each item of the array shall have as value the unique identifier of one signature validation policy against which the server is able to validate digital signatures, as an URI. If the identifier of the signature validation policy is an OID, then the value of this element shall be an URN indicating the value of the aforementioned OID as specified in IETF RFC 3061 [14].

5.2.3.2.5 Component for returning the detailed validation report (signed or unsigned)

5.2.3.2.5.1 Component semantics

This component shall contain either the signed or the unsigned detailed validation report for one or more digital signatures that the server has validated in response to one request that requested its generation (incorporating the component specified in clause 5.1.4.2.3 of the present document).

If this element is child of the signature processing results container it shall contain the validation report of one signature (the one that the signature processing results container provides details of).

If this element is child of the optional outputs component, it may contain the validation report on more than one signatures.

Clause 8.4.3.3 provides details on what the server may decide regarding the generation of this component when building the validation response message.

When the validation report is used for reporting on the validation of more than one signature, its contents shall contain mechanisms for identifying each one of the validated and reported signatures.

NOTE: The format defined in ETSI TS 119 102-2 [11] for the validation report allows reporting on the validation of several signatures and includes mechanisms for identifying the reported signatures.

5.2.3.2.5.2 XML component

The element that shall contain the detailed validation report for one or more digital signatures shall be the `ValidationReport`, which shall be an instance of `etsival:ValidationReportContainerType` defined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1, and is copied below for information.

```
<!—targetNamespace=http://uri.etsi.org/19442/v1.1.1# -->

<xs:element name="ValidationReport" type="etsival:ValidationReportContainerType"/>

<xs:complexType name="ValidationReportContainerType">
    <xs:choice>
            <xs:element name="ETSITS11910202XMLReport" type="etsivr:ValidationReportType"
minOccurs="0"/>
            <xs:element name="other" type="xs:base64Binary" minOccurs="0"/>
    </xs:choice>
    <xs:attribute name="isSigned" type="xs:boolean" use="required"/>
    <xs:attribute name="Encoding" type="xs:string" use="optional"/>
    <xs:attribute name="SpecificationId" type="xs:anyURI" use="optional"/>
</xs:complexType>
```

The `ETSITS11910202XMLReport` element shall contain an instance of the XML validation report specified in ETSI TS 119 102-2 [11].

The `isSigned` attribute shall indicate whether the validation report is signed or not.

The `other` element is a container for validation reports different than the XML validation report specified in ETSI TS 119 102-2 [11].

Attributes `Encoding` and `SpecificationId` shall not be present if the `ETSITS11910202XMLReport` element is present. They may be present only if the `other` element is present.

Attribute `Encoding` shall indicate the encoding used for the validation report present in other element.

The value of attribute `SpecificationId` shall be an URI identifying the specification where the validation report present in `other` is defined.

5.2.3.2.5.3 JSON component

The element that shall contain the detailed validation report for one or more digital signatures shall be the `validationReport`. This value of this element is the base-64 encoding of a signed or unsigned `etsivr:ValidationReport` XML element as specified in the XML Schema of ETSI TS 119 102-2 [11].

The `validationReport` element shall be an instance of `ValReportContainerType` defined as in JSON Schema file "19442jsonSchema.json", whose location is detailed in clause A.2, and is copied below for information.

```
"ValReportContainerType":{
  "type": "object",
  "properties": {
    "etsiTS11910202XMLReport":{
      "type": "string",
      "contentEncoding": "base64"
    },
    "other":{
      "type": "object",
      "properties": {
        "content": {
          "type": "string",
          "contentEncoding": "base64"
        },
        "specId": {"type": "string"},
        "encoding": {"type": "string"}
      }
    },
    "isSigned": {"type": "boolean"}
  }
}
```

The `etsiTS11910202XMLReport` element shall contain the base-64 encoding of an instance of the XML validation report specified in ETSI TS 119 102-2 [11].

The `isSigned` component shall indicate whether the validation report is signed or not.

The `other` component is a container for validation reports different than the XML validation report specified in ETSI TS 119 102-2 [11].

Component `encoding` shall indicate the encoding used for the validation report present in the `other` element.

The value of attribute `specId` shall be an URI identifying the specification where the validation report present in `other` is defined.

5.2.3.3 Components re-used from DSS-X core v2.0

5.2.3.3.1 Component for indicating the applied service policy

5.2.3.3.1.1 Component semantics

This component shall contain a non-ambiguous identifier of the service policy under which the client requests the server to validate the signature.

5.2.3.3.1.2 XML component

The element that shall identify under which service policy the validation has to be conducted shall be the `dsb:AppliedPolicy` element specified in clause 4.1.9.2 of DSS-X core v2.0 [1].

5.2.3.3.1.3 JSON component

The element that shall identify under which service policy the validation has to be conducted shall be the `policy` element. The contents of this element shall be as specified in clause 4.1.9.1 of DSS-X core v2.0 [1].

5.2.3.3.2 Component for indicating validation time

5.2.3.3.2.1 Component semantics

This component shall provide means for indicating to the client the validation time set by the server, which may be the current time or a certain time in the past.

This component shall appear in the response to requests of validation of signatures that incorporate the component requesting the server to set the validation time at a certain time instant, specified in clause 5.1.4.3.5.1 of the present document.

5.2.3.3.2.2 XML component

The element that shall report the validation time used by the server shall be the `dss2:VerificationTimeInfo` element specified in clause 4.3.27.2 of DSS-X core v2.0 [1].

5.2.3.3.2.3 JSON component

The element that shall report the validation time used by the server shall be the element `verificationTimeInfo` an instance of `dss2-VerificationTimeInfoType` type specified in clause 4.3.27.1 of DSS-X core v2.0 [1].

5.2.3.3.3 Component for returning the signing time of one signature

5.2.3.3.3.1 Component semantics

This component shall provide means for indicating to the client information on the signing time.

This component shall appear in the response to requests of validation of signatures that incorporate the component requesting the server to return information of signing time, specified in clause 5.1.4.3.8.1 of the present document.

5.2.3.3.3.2 XML component

The element that shall convey information on signing time shall be the `dss2:SigningTimeInfo` element specified in clause 4.3.31.2 of DSS-X core v2.0 [1].

5.2.3.3.3.3 JSON component

The element that shall convey information on signing time shall be the element `signingTimeInfo` an instance of `dss2-SigningTimeInfoType` type specified in clause 4.3.31.1 of DSS-X core v2.0 [1].

5.2.3.3.4 Component for returning signer's identity

5.2.3.3.4.1 Component semantics

This component of a response to a validation request shall return information on the signer's identity.

This component shall appear in the response to requests that incorporate the component requesting the server to return these details, specified in clause 5.1.4.3.9.1 of the present document.

5.2.3.3.4.2 XML component

The element that shall contain the identity of the signer shall be the `dss2:SignerIdentity` element, specified in clause 4.3.8.2 of DSS-X core v2.0 [1].

5.2.3.3.4.3 JSON component

The element that shall contain the identity of the signer shall be the `signerIdentity` element, whose contents shall be as specified in clauses 4.3.8.1 and 4.4.1.1 of DSS-X core v2.0 [1].

5.2.3.3.5 Component for returning the result of transforming the input document

5.2.3.3.5.1 Component semantics

This component shall appear only in the response to requests of validation of XAdES signatures that incorporate the component requesting the server to return one or more transformed input documents, specified in clause 5.1.4.3.10.1 of the present document.

This component shall provide means for returning to the client the result obtained by the server after applying to one input document signed by a XAdES signature, the sequence of transformations indicated in a certain `ds:Reference` element of that signature referencing such input document.

5.2.3.3.5.2 XML component

The element that shall return the the result obtained by the server after applying a sequence of transformations to one input document shall be the `dss2:TransformedDocument` of type `dss2:TransformedDocumentType specified` in clause 4.3.34.2 of DSS-X core v2.0 [1].

5.2.3.3.5.3 JSON component

The element that shall return the result obtained by the server after applying a sequence of transformations to one input document shall be the `transformed` element of type of `dss2-TransformedDocumentType` specified in clause 4.3.34.1 of DSS-X core v2.0 [1].

5.2.3.3.6 Component for returning the result of validating ds:Manifest elements in XAdES signatures

5.2.3.3.6.1 Component semantics

This component shall appear only in the response to requests of validation of XAdES signatures that incorporate the component requesting the server to validate signed `ds:Manifest` elements.

This component shall contain the result of the validation(s) performed by the server on the signed `ds:Manifest` elements present within a XAdES signature.

5.2.3.3.6.2 XML component

The element that shall contain the result of the validation(s) performed by the server on signed the `ds:Manifest` elements present within the XAdES signature(s) shall be the `dss2:VerifyManifestResults` element, specified in clause 4.3.23.2 of DSS-X core v2.0 [1].

5.2.3.3.6.3 JSON component

The element that shall contain the result of the validation(s) performed by the server on the signed `ds:Manifest` elements present within the XAdES signature(s) shall be the `manifestValResult` element. This element shall be an instance of `dss2-VerifyManifestResultsType` type specified in clause 4.3.23.1 of DSS-X core v2.0 [1].

6 Protocol for augmentation of AdES signatures

6.1 Request message

6.1.1 Component for requesting augmentation of signatures

6.1.1.1 Component semantics

The signature augmentation request message shall allow requesting to the server the augmentation of one or more AdES signatures.

The request message shall be able to submit the signatures to be augmented, following the same principles as in the validation request message, either:

1) within the signature object container (if the signature is a non-embedded signature); or
2) within the input documents container (if the signature is an embedded signature); or
3) within an underlying protocol attachment (if the signature is an embedded signature), in which case this attachment shall be referenced within the input documents container.

The request message shall not include neither transformed documents nor digests of signed documents.

The request message may incorporate documents.

The request message shall include a set of additional input components for detailing what is actually requested to the server.

The request message shall allow to request augmenting different signatures to the same level. It shall not support requesting to augment different signatures to different levels.

The request message may contain a component whose value is an identifier of the message itself.

The augmentation request message shall contain one or more components identifying protocols and/or profiles that the response message is compliant with. The first one of such components shall have the following URI as value, identifying the response as one that has been built using the "augmentation" protocol specified in the present document:

- `http://uri.etsi.org/19442/v1.1.1/augmentationprotocol#`.

6.1.1.2 XML component

The element that shall be the component for requesting the augmentation of AdES signature(s) shall be the root element of the message `AugmentRequest` as specified in the present clause.

The `AugmentRequest` element shall be defined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1, and is copied below for information.

```
<!—targetNamespace="http://uri.etsi.org/19442/v1.1.1#" →

    <xs:element name="AugmentRequest" type="etsival:AugmentRequestType"/>

    <xs:complexType name="AugmentRequestType">
        <xs:complexContent>
            <xs:extension base="dss2:RequestBaseType">
                <xs:sequence>
                    <xs:element ref="dss2:InputDocuments" minOccurs="0"/>
                    <xs:element name="AdditionalInputs" type="dss2:OptionalInputsVerifyType"/>
                    <xs:element ref="dss2:SignatureObject" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

The `AugmentRequest` element shall have one or more `dsb:Profile` children elements. The first one shall have the value `http://uri.etsi.org/19442/v1.1.1/augmentationprotocol#`, identifying the request as an augmentation request compliant with the augmentation protocol specified in the present document.

The `dss2:InputDocuments` element shall only contain one or more `dss2:Document` children elements. These children elements shall contain one or more embedded AdES signatures or references to underlying protocol attachments where the documents with embedded AdES signatures.

The `dss2:SignatureObject` child element shall not contain any time-stamp token.

6.1.1.3 JSON component

The element that shall be the component for requesting the validation of AdES signature(s) shall be the root element of the message `AugmentRequest` as specified in the present clause.

The `AugmentRequest` element shall be defined as in JSON Schema file "19442jsonSchema.json, whose location is detailed in clause A.2, and is copied below for information.

```
"AugmentRequest": {
    "type": "object",
    "properties": {
        "inDocs": {"$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-InputDocumentsType"},
        "reqID": {
            "type": "string"
        },
        "profile": {
            "type": "array",
            "items": {
                "type": "string",
                "format": "uri"
            }
        },
        "claimedIdentity": {
            "$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-ClaimedIdentityType"
        },
        "returnAugmentedSig": {
            "type": "string"
            "format": "uri"
        },
        "processSigs": {
            "$ref": "#/definitions/SigsRefsType"
        },
        "sigObj": {
            "$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-SignatureObjectType"
        },
        "addKeyInfo": {
            "type": "array",
            "items": {"$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-AdditionalKeyInfoType"}
        },
        "tstksQualityLevel": {
            "type": "string"
            "format": "uri"
        }
    },
    "required": ["profile","returnAugmentedSig"]
}
```

The `profile` array shall have one or more items. The first one shall have the value `http://uri.etsi.org/19442/v1.1.1/augmentationprotocol#`, identifying the request as an augmentation request compliant with this augmentation protocol.

The `inDocs` element shall only contain one or more `doc` children elements. These children elements shall contain one or more embedded AdES signatures or references to underlying protocol attachments where the documents with embedded AdES signatures.

The `claimedIdentity` element shall contain details of the identity claimed by the client. The requirements specified in clause 5.1.4.3.3.3 of the present document shall apply to this element.

The `returnAugmentedSig` element shall indicate the level to which the client requests to augment the signatures. The requirements specified in clause 6.1.2.2.3 of the present document shall apply to this element.

The `sigObj` child element shall not contain any time-stamp token.

The `processSigs` element shall reference the signatures to be augmented. The requirements specified in clause 5.1.4.2.1.4 of the present document shall apply to this element.

The `addKeyInfo` child element shall contain validation material for the signatures to be augmented. The requirements specified in clause 5.1.4.3.7.1 of the present document shall apply to this element.

The `tstksQualityLevel` element shall identify the level of quality of the time-stamp tokens used for augmenting the signatures, if time-stamp tokens are used. The requirements specified in clause 6.1.2.2.3 of the present document shall apply to this element.

6.1.2 Additional inputs

6.1.2.1 Container for additional inputs

6.1.2.1.1 Semantics

This container shall include:

1) One component for indicating the level that the submitted signature(s) has(have) to be augmented to. Clause 6.1.2.2 defines requirements for this component.

In addition, this container may also include:

1) One component for identifying the signatures that the server is requested to augment. Its requirements are as specified in clause 5.1.4.2.1.
2) One component for allowing the client to claim its identity. Its requirements are as specified in clause 5.1.4.3.3.
3) One component for passing to the server validation material for the submitted signature(s). Its requirements are as specified in clause 5.1.4.3.7.
4) One component for requesting a certain level of quality for time-stamp tokens if the augmentation of the signature(s) require time-stamp tokens. Clause 6.1.2.3 defines requirements for this component.

6.1.2.1.2 XML component

The `AdditionalInputs` element shall be the container for additional inputs submitted to the server. It shall be an instance of `dss2:OptionalInputsVerifyType` type, suitably redefined in clause 5.1.4.1.2 of the present document.

This element shall not contain the following children:

- `dsb:Other`.
- `dss2:AddTimeStamp`.
- `dss2:UseVerificationTime`.
- `dss2:ReturVerificationTimeInfo`.
- `dss2:ReturnProcessingDetails`.
- `dss2:ReturnSigningTimeInfo`.
- `dss2:ReturnSignerIdentity`.
- `dss2:ReturnTransformedDocument`.

- `dss2:ReturnTimeStampedSignature`.
- `dss2:VerifyManifests`.
- `UseSignatureValidationPolicy`.
- `ReturnValidationReport`.
- `ProofsOfExistence`.

The `ReturnAugmentedSignature` element shall always be present. It shall indicate the level to which the client requests to augment the signatures. The requirements for this component are given in clause 6.1.2.2.2 of the present document.

The `ProcessSignatures` element may be present. It shall reference the signatures to be augmented. The requirements specified in clause 5.1.4.2.1.3 of the present document shall apply to this element.

The `dss2:ClaimedIdentity` child element may be present. It shall contain validation material for the signatures to be augmented. The requirements specified in clause 4.3.9.2 of DSS-X core v2.0 [1] shall apply to this element.

The `dss2:AdditionalKeyInfo` child element may be present. It shall contain validation material for the signatures to be augmented. The requirements specified in clause 5.1.4.3.7.2 of the present document shall apply to this element.

The `TSTokensQualityLevel` element may be present. It shall identify the level of quality of the time-stamp tokens used for augmenting the signatures, if time-stamp tokens are used. The requirements for this component are given in clause 6.1.2.3.2 of the present document.

6.1.2.1.3 JSON component

In the JSON protocol the additional components appear as direct children of the root node of the message as shown in clause 6.1.1.3 of the present document.

6.1.2.2 Component for identifying the level the signatures are requested to be augmented to

6.1.2.2.1 Component semantics

This component shall have as value a URI reference identifying the pre-defined level to which the server is requested to augment the signature.

The value of this component shall take one of the values in Table 5 and Table 6 except the URIs identifying levels AdES-B-B, AdES-E-BES, AdES-E-EPES, AdES-B, AdES-BES, and AdES-EPES.

NOTE 1: The levels mentioned in the former paragraph are not the result of an augmentation operation, but the result of the actual generation of the AdES signature.

Table 5 lists the URIs for the levels specified for AdES signatures in ETSI EN 319 122 [2], ETSI EN 319 132 [3], and ETSI EN 319 142 [4]. Numbers in column Notes correspond to the numbers of the notes that follow the table.

Table 5 : AdES signature levels in ETSI EN 319 1X2 and URIs

Signature level	URI	Notes
AdES-B-B	`http://uri.etsi.org/ades/191x2/level/baseline/B-B#`	2
AdES-B-T	`http://uri.etsi.org/ades/191x2/level/baseline/B-T#`	2
AdES-B-LT	`http://uri.etsi.org/ades/191x2/level/baseline/B-LT#`	2
AdES-B-LTA	`http://uri.etsi.org/ades/191x2/level/baseline/B-LTA#`	2
AdES-E-BES	`http://uri.etsi.org/ades/191x2/level/extended/E-BES#`	2
AdES-E-EPES	`http://uri.etsi.org/ades/191x2/level/extended/E-EPES#`	2
AdES-E-T	`http://uri.etsi.org/ades/191x2/level/extended/E-T#`	2
AdES-E-C	`http://uri.etsi.org/ades/191x2/level/extended/E-C#`	3
AdES-E-X	`http://uri.etsi.org/ades/191x2/level/extended/E-X#`	3
AdES-E-X-Long	`http://uri.etsi.org/ades/191x2/level/extended/E-X-Long#`	3
AdES-E-X-L	`http://uri.etsi.org/ades/191x2/level/extended/E-X-L#`	3
AdES-E-A	`http://uri.etsi.org/ades/191x2/level/extended/E-A#`	3
AdES-E-LTV	`http://uri.etsi.org/ades/191x2/level/extended/E-LTV#`	4
CAdES-E-ERS	`http://uri.etsi.org/ades/191x2/level/extended/E-ERS`	5

NOTE 2: The levels identified in these rows are levels that CAdES, PAdES, and XAdES signatures can reach.

NOTE 3: The levels identified in these rows are levels that CAdES and XAdES signatures can reach, but not PAdES signatures.

NOTE 4: The levels identified in these rows are levels that only PAdES signatures can reach.

NOTE 5: At the moment of producing the present document, only CAdES signatures defined levels with Evidence Records.

Table 6 lists the URIs for the levels specified for AdES signatures in the different ETSI TSs, namely ETSI TS 101 733 [5], ETSI TS 102 778 [9], ETSI TS 101 903 [7], ETSI TS 103 171 [8], ETSI TS 103 172 [10], and ETSI TS 103 173 [6]. Numbers in column Notes correspond to the numbers of the notes that follow the table.

Table 6: AdES signature levels in ETSI TSs and URIs

Signature level	URI	Notes
AdES-B	`http://uri.etsi.org/ades/etsits/level/baseline/B-B#`	5
AdES-T	`http://uri.etsi.org/ades/etsits/level/baseline/B-T#`	5
AdES-LT	`http://uri.etsi.org/ades/etsits/level/baseline/B-LT#`	5
AdES-LTA	`http://uri.etsi.org/ades/etsits/level/baseline/B-LTA#`	5
AdES-BES	`http://uri.etsi.org/ades/etsits/level/BES#`	5
AdES-EPES	`http://uri.etsi.org/ades/etsits/level/EPES#`	5
AdES-T	`http://uri.etsi.org/ades/etsits/level/T#`	5
AdES-C	`http://uri.etsi.org/ades/etsits/level/C#`	6
AdES-X	`http://uri.etsi.org/ades/etsits/level/X#`	6
AdES-X-Long	`http://uri.etsi.org/ades/etsits/level/X-Long#`	6
AdES-X-L	`http://uri.etsi.org/ades/etsits/level/X-L#`	6
AdES-A	`http://uri.etsi.org/ades/etsits/level/A#`	6
AdES-LTV	`http://uri.etsi.org/ades/etsits/level/LTV#`	7

NOTE 6: The levels identified in these rows are levels that CAdES, PAdES, and XAdES signatures can reach.

NOTE 7: The levels identified in these rows are levels that CAdES and XAdES signatures can reach, but not PAdES signatures.

NOTE 8: The levels identified in these rows are levels that only PAdES signatures can reach.

6.1.2.2.2 XML component

The element that shall request the augmenting of the signature shall be the `ReturnAugmentedSignature` element.

The `ReturnAugmentedSignature` element shall be defined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1, and is copied below for information.

```
<!—targetNamespace="http://uri.etsi.org/19442/v1.1.1#" →

    <xs:element name="ReturnAugmentedSignature" type="etsival:ReturnAugmentedSignatureType"/>

    <xs:complexType name="ReturnAugmentedSignatureType">
        <xs:attribute name="Level" type="xs:anyURI" use="required"/>
    </xs:complexType>
```

The `Level` attribute of the aforementioned element shall have the value indicating the level to which the server is requested to augment the signatures. This attribute shall take one of the values in Table 5 and Table 6 except the URIs identifying levels AdES-B-B, AdES-E-BES, AdES-E-EPES, AdES-B, AdES-BES, and AdES-EPES.

6.1.2.2.3 JSON component

The element that shall request the augmenting of the signature after its validation shall be the `returnAugmentedSig` component.

This component shall have the value indicating the level to which the server is requested to augment the signatures.

This component shall take one of the values in Table 5 and Table 6 except the URIs identifying levels AdES-B-B, AdES-E-BES, AdES-E-EPES, AdES-B, AdES-BES, and AdES-EPES.

6.1.2.3 Component for identifying the quality level of the time-stamp tokens used in the augmentation process

6.1.2.3.1 Component semantics

This component shall allow to identify the quality level that time-stamp tokens participating in the augmentation of the signature shall have.

6.1.2.3.2 XML component

The element that shall request the quality level of the time-stamp tokens used for augmenting the signature shall be the `TSTokensQualityLevel` element.

The `TSTokensQualityLevel` element shall be defined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1, and is copied in clause 4.3.2 of the present document.

6.1.2.3.3 JSON component

The element that shall request the quality level of the time-stamp tokens used for augmenting the signature shall be the `tstksQualityLevel` element.

The `tstkQualityLevel` element shall be defined as in JSON Schema file "19442jsonSchema.json", whose location is detailed in clause A.2, and is copied in clause 6.1.1.3 of the present document.

6.2 Response message

6.2.1 Component for responding to augmentation request

6.2.1.1 Component semantics

The signature augmentation response message shall allow the server to return to the client one or more AdES signatures, which shall be the result of augmenting the AdES signatures submitted by the client to the requested level.

The signature augmentation response message shall have one or more signature result containers for returning either:

1) the augmented signature; or

2) an indication of error while trying to augment the signature and a reference to the non augmented signature.

The response message shall be able to return the augmented signatures either:

1) within the signature object container (if the signature is a non-embedded signature); or

2) within the document with signature container (if the signature is an embedded signature); or

3) within an underlying protocol attachment (if the signature is an embedded signature), in which case this attachment shall be referenced within the document with signature container.

NOTE: In essence, an augmentation response message needs to submit only the augmented AdES signatures. However, if these are embedded signatures, the message can submit the signed document(s) with the embedded signatures.

This message shall contain one or more components identifying protocols and/or profiles that the response message is compliant with. The first one of such components shall have the following URI as value, identifying the response as one that has been built using the augmentation protocol specified in the present document:

- `http://uri.etsi.org/19442/v1.1.1/augmentationprotocol#`

This message may contain a component whose value is an identifier of the message itself.

This message may contain component whose value is the identifier of the request message that this message is the response of.

6.2.1.2 XML component

The element that shall be the component for responding to a request of augmentation of AdES signature(s) shall be the root element of the message `AugmentResponse` as specified in the present clause.

The `AugmentResponse` element shall be defined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1, and is copied below for information.

```
<!—targetNamespace="http://uri.etsi.org/19442/v1.1.1#" →

    <xs:element name="AugmentResponse" type="etsival:AugmentResponseType"/>

    <xs:complexType name="AugmentResponseType">
        <xs:complexContent>
            <xs:extension base="dsb:ResponseBaseType">
                <xs:sequence>
                    <xs:element ref="etsival:AugmentSignatureResult" maxOccurs="unbounded"/>
                    <xs:element ref="dss2:DocumentWithSignature"minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
```

The `AugmentResponse` element shall have one or more `dsb:AppliedProfile` children. The first one shall have the value `http://uri.etsi.org/19442/v1.1.1/augmentationprotocol#`, identifying the request as an augmentation response compliant with the augmentation protocol specified in the present document.

There shall be as many `AugmentSignatureResult` elements as signatures the client requested to augment.

Each `AugmentSignatureResult` element shall contain the details of the process performed by the server for augmenting one AdES signature present in the request. Among other children, this element shall include either the augmented signature, if this is not embedded, or a pointer to one signature embedded in one of the documents encapsulated by one `dss2:DocumentWithSignature` element.

There shall be as many `dss2:DocumentWithSignature` elements as input documents embedding the signatures that the client requested to augment.

Each `dss2:DocumentWithSignature` element shall contain one document enveloping one or more augmented signatures. More than one `AugmentedSignatureResult` elements may reference different augmented signatures embedded within one `dss2:DocumentWithSignature` element.

6.2.1.3 JSON component

The element that shall be the component for responding to the validation request of AdES signature(s) shall be the root element of the message `AugmentResponse` as specified in the present clause.

The `AugmentResponse` element shall be an instance of `AugmentResponseType`, defined as in JSON Schema file "19442jsonSchema.json, whose location is detailed in clause A.2, and is copied below for information.

```
"AugmentResponseType": {
    "type": "object",
    "properties": {
        "result": {
            "$ref": "<DSSXBASESCHEMAFILELOCATION>#/definitions/dsb-ResultType"
        },
        "reqID": {
            "type": "string"
        },
        "profile": {
            "type": "array",
            "items": {
                "type": "string"
            }
        },
        "augmentSigsResults": {
            "type": "array",
            "items": {
                "$ref": "#/definitions/AugmentSigResultType"
            }
        }
        "docsWithSignatures": {
            "type": "array",
            "items": {
                "$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-DocumentWithSignatureType"
            }
        }
    },
    "required": ["result", "profile"]
}
```

The `profile` array shall have one or more items. The value of the first one shall be `http://uri.etsi.org/19442/v1.1.1/augmentationprotocol#`, identifying the response as an augmentation response compliant with the validation protocol specified in the present document.

There shall be as many items within `augmentSigsResult` array as signatures the client requested to augment.

Each item in `augmentSigsResult` array shall contain the details of the process performed by the server for augmenting one AdES signature present in the request. Among other components, this element shall include either the augmented signature, if this is not embedded, or a pointer to one signature embedded in one of the documents encapsulated by one of the items in `docsWithSignatures` element.

There shall be as many items within `docsWithSignatures` element as input documents embedding the signatures that the client requested to augment.

Each item in `docsWithSignatures` element shall contain one document enveloping one or more augmented signatures. More than one item in `augmentSigsResult` element may reference different augmented signatures embedded within one item in `docsWithSignatures` element.

6.2.2 Component for the global augmentation result

6.2.2.1 Component semantics

This component shall contain a major result, which shall report whether the protocol has been successfully executed, regardless all, or only some, or none of the signatures whose augmentation was requested, have actually been augmented or not.

For details of the values of its result major and result minor, see clause 8.4.3.1 of the present document.

6.2.2.2 XML component

The element that within the response shall notify the validation result shall be the `dsb:Result` element specified in clause 4.1.7.2 of DSS-X core v2.0 [1].

6.2.2.3 JSON component

The element that shall notify the global result shall be the `result` component of type `dsb-ResultType` specified in clause 4.1.7.1 of DSS-X core v2.0 [1].

6.2.3 Augment signature result container

6.2.3.1 Component semantics

This component shall include an element referencing the signature within the `AugmentRequest` message whose augmentation process details it reports.

This component shall include an element for notifying the result of the augmentation process of the aforementioned referenced signature. Its contents shall be as specified in clause 8.4.3.6 of the present document.

This component may also include an element that allows to contain either:

- the augmented signature if it is a non-embedded signature (in this case the component shall meet the requirements specified in clause 4.3.32 of DSS-X core v2.0 [1]; or
- a reference to the augmented signature if it is an embedded signature. If the embedded and augmented signature is a PAdES signature, the reference shall include the name of the PDF field where the signature is placed within the PDF file.

See clause 8.4.3.6 for details of the processing model for building this component.

6.2.3.2 XML component

The component for reporting on the augmentation of one AdES signature shall be an the element `AugmentSignatureResult` as specified in the present clause.

The `AugmentSignatureResult` element shall be defined as in XML Schema file "19442xmlSchema.xsd", whose location is detailed in clause A.1, and is copied below for information.

```
<!—targetNamespace="http://uri.etsi.org/19442/v1.1.1#" →

    <xs:element name="AugmentSignatureResult" type="etsival:AugmentSignatureResultType"/>

    <xs:complexType name="AugmentSignatureResultType">
        <xs:sequence>
            <xs:element ref="dsb:Result" />
            <xs:element name="SignatureRefInRequest" type="etsivr:SignatureReferenceType"
minOccurs="0" />
```

```
            <xs:element ref="dss2:AugmentedSignature" minOccurs="0"/>
            <xs:element name="PAdESFieldName" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
```

The `SignatureRefInRequest` element shall be present in this protocol.

The `dss2:AugmentedSignature` contains either the augmented signature if it is non-embedded, or a pointer to an augmented XAdES signature embedded within a XML document enclosed within a `DocumentWithSignature` element.

The `PAdESFieldName` element shall indicate the PDF field name within the PDF document enclosed in a `dss2:DocumentWithSignature` element where the PAdES signature is placed.

6.2.3.3 JSON component

The component for reporting on the augmentation of one AdES signature shall be each of the items of `augmentSigsResults` array. Each item shall be an instance of `AugmentSigResultType` type.

The `AugmentSigResultType` type shall be defined as in JSON Schema file "19442jsonSchema.json", whose location is detailed in clause A.1, and is copied below for information.

```
"AugmentSigResultType": {
    "type": "object",
    "properties": {
        "result": {
            "$ref": "<DSSXBASESCHEMAFILELOCATION>#/definitions/dsb-ResultType"
        },
        "augmentedSig": {
            "$ref": "http://docs.oasis-open.org/dss-x/dss-
core/v2.0/csprd01/schema/schema.json#definitions/dss2-AugmentedSignatureType"
        },
        "sigRefInReq": {
            "$ref": "#/definitions/SignatureReferenceType"
        }
    },
    "required": ["result", "sigRef"]
}
```

The `sigRefInReq` component shall be present in this protocol. It shall be an instance of the `SignatureReferenceType` type defined in clause 5.1.4.2.4.3 of the present document.

The `augmentedSig` contains either the augmented signature if it is non-embedded, or a pointer to an augmented XAdES signature embedded within a XML document enclosed within one of the items of the `docsWithSignatures` array. It shall be an instance of `dss2-AugmentedSignatureType` type as specified in clause 4.3.32.1 of DSS-X core v2.0 [1].

The `PAdESFieldName` element shall indicate the PDF field name within the PDF document enclosed in the first and unique item within `docsWithSignatures` array where the PAdES signature is placed.

7 Protocol for validation and augmentation of AdES signatures

7.1 Request message

7.1.1 Component for requesting validation and augmentation

7.1.1.1 Component semantics

The requirements for this component shall be the requirements specified in clause 5.1.1.1 except for the following difference:

- This component shall have a component identifying one or more protocols and/or profiles that the response message is compliant with. The first one of such components shall have the following URI as value, identifying the request message as one that has been built using the validation and augmentation protocol specified in the present document:
  `http://uri.etsi.org/19442/v1.1.1/validationAndAugmentationprotocol#`.

7.1.1.2 XML component

The element that shall be the component for requesting the validation and augmentation of AdES signature(s) shall be the root element of the message `dss2:VerifyRequest` as specified in clause 5.1.1.2 with the following difference:

1) the `dss2:VerifyRequest` element shall have one or more `dsb:Profile` children elements. The first one shall have the value `http://uri.etsi.org/19442/v1.1.1/validationAndAugmentationprotocol#`, identifying the request as a validation and augmentation request compliant with the validation and augmentation protocol specified in the present document.

7.1.1.3 JSON component

The element that shall be the component for requesting the validation and augmentation of AdES signature(s) shall be the root element of the message `VerifyReq` as specified in clause 5.1.1.3 of the present document with the following difference:

1) the `profile` array shall have one or more items. The first one shall have the value `http://uri.etsi.org/19442/v1.1.1/validationAndAugmentationprotocol#`, identifying the request as a validation and augmentation request compliant with the validation and augmentation protocol specified in the present document.

7.1.2 Components for submitting signatures and signed documents

The components for submitting signatures and signed documents shall be the same as the components used for submitting signatures and signed documents in the request messages of the validation protocol.

Requirements in clause 5.1.2.1 apply for submitting the signatures to be validated and augmented. The XML elements used for submitting the signatures shall be the ones specified in clause 5.1.2.2. The JSON elements for submitting the signatures shall be the ones specified in clause 5.1.2.3.

Requirements in clause 5.1.3.1 apply for submitting the documents signed by the signatures. The XML elements used for submitting the documents shall be the ones specified in clause 5.1.3.2. The JSON elements used for submitting the documents shall be the ones specified in clause 5.1.3.3.

7.1.3 Optional components

7.1.3.1 Container for optional components

7.1.3.1.1 Component semantics

The requirements in clause 5.1.4.1.1 apply to this component, except for the following difference:

1) This component shall include a component for requesting the augmentation of the signatures to a certain level.

NOTE: Despite the fact that this component is mandatory for the validation and augmentation protocol, it appears within the group of optional inputs, which are not optional any more, for keeping as much alignment with DSS-X OASIS protocol, which does not make the separation between "validation" protocol and "validation and augmentation" protocol.

7.1.3.1.2 XML component

The container for the optional components shall be an instance of the `OptionalInputsVerifyType` type specified in clause 5.1.4.1.2.

All the requirements of clause 5.1.4.1.2 shall apply in the validation and augmentation request message with the following differences:

1) The `ReturnAugmentedSignature` element shall be present.

2) The `TSTokensQualityLevel` element may be present.

7.1.3.1.3 JSON component

The container for the optional components shall be an instance of the `OptionalInputsVerifyType` type specified in clause 5.1.4.1.3.

All the requirements of clause 5.1.4.1.3 shall apply in the validation and augmentation request message with the following differences:

1) The `returnAugmentedSig` element shall be present.

2) The `tstkQualityLevel` element may be present.

7.2 Response message

7.2.1 Component for responding to validation and augmentation request

7.2.1.1 Component semantics

The requirements for this component shall be the requirements specified in clause 5.2.1.1, except by the following differences:

1) This component shall have a profile identifier that shall identify the message as compliant with the "validation and augmentation" protocol specified in the present document.

7.2.1.2 XML component

The element that shall be the component for responding to a request of validation and augmentation of AdES signature(s) shall be the root element of the message `dss2:VerifyResponse` as specified in clause 5.2.1.2 with the following difference:

1) the dss2:VerifyResponse element shall have one or more `dsb:Profile` children elements. The first one shall have the value `http://uri.etsi.org/19442/v1.1.1/validationAndAugmentationprotocol#`, identifying the response as a validation and augmentation response compliant with the augmentation and validation protocol specified in the present document.

7.2.1.3 JSON component

The element that shall be the component for responding to a request of validation and augmentation of AdES signature(s) shall be an instance of `VerifyResp` as specified in clause 5.2.1.3 of the present document with the following difference:

1) the `profile` array shall have one or more items. The first one shall have the value `http://uri.etsi.org/19442/v1.1.1/validationAndAugmentationprotocol#`, identifying the response as a validation and augmentation response compliant with the validation and augmentation protocol specified in the present document.

7.2.2 Component for the global validation and augmentation result

7.2.2.1 Component semantics

This component shall contain a major result, which shall report whether the server has been able to perform its task, regardless the results obtained. This component may also contain a minor result providing additional information on the task performed by the server.

For details of the values of its result major and result minor, see clause 8.4.3.1 of the present document.

7.2.2.2 XML component

The element that shall notify the global validation and augmentation result shall be the `dsb:Result` element specified in clause 4.1.7.2 of DSS-X core v2.0 [1].

7.2.2.3 JSON component

The element that shall notify the global validation and augmentation result shall be `result` of type `dsb-ResultType` type specified in clause 4.1.7.1 of DSS-X core v2.0 [1].

7.2.3 Optional components

7.2.3.1 Container for optional components

7.2.3.1.1 Component semantics

The requirements in clause 5.2.3.1.1 apply to the component consisting in a sequence of signature results container.

7.2.3.1.2 XML component

The element for incorporating the sequence of signature result containers shall be an instance of `OptionalOutputsVerifyType` type specified in clause 5.2.3.1.2. The requirements specified in clause 5.2.3.1.2 shall apply for this element.

7.2.3.1.3 JSON component

The element for incorporating the sequence of signature result containers shall be the `oplOutp` property, instance of `OptionalOutputsVerifyType` type as specified in clause 5.2.3.1.3. The requirements specified in clause 5.2.3.1.3 shall apply for this element.

7.2.3.2 Signature processing results container

7.2.3.2.1 Component semantics

The requirements in clause 5.2.3.2.1 shall apply to this component with the following difference:

1) It shall include the augment signature result container.

7.2.3.2.2 XML component

The element for implementing the signature results container shall be the `ResultsForOneSignature` element as specified in clause 5.2.3.2.1.2. The requirements specified in clause 5.2.3.2.1.2 shall apply for this element with the following differences:

1) This element shall include the `AugmentSignatureResult` element.

2) The `AugmentSignatureResult` element shall not include the `SignatureRefInRequest` element, as the `SignatureReference` child element of `ResultsForOneSignature` element already contains the reference to the validated and augmented signature.

7.2.3.2.3 JSON component

The element for implementing the signature results container shall be the `resForOneSig` property as specified in clause 5.2.3.2.1.3. The requirements specified in the aforementioned clause shall apply for this element with the following differences:

1) This element shall include the `augmentSigResult` child.

2) The `augmentSigResult` element shall not include the `SigRefInReq` child, as the `SignatureReference` child element of `resForOneSig` element already contains the reference to the validated and augmented signature.

7.2.4 Reporting results

7.2.4.1 Introduction

Being the validation and the attempt to augment one signature two different processes, their results shall be reported in two different result components in the response message, as indicated below.

7.2.4.2 Reporting results in XML protocol

The result of the validation of a certain signature shall be reported in the `dsb:Result` child element of the `ResultsForOneSignature` element reporting the processing results for that signature. The contents of this element shall be as specified in clause 8.4.3.1 of the present document.

The result of the attempt to augment a certain signature shall be reported in the `dsb:Result` child element of the `AugmentSignatureResult` element reporting the results of that augmentation attempt. The contents of this element shall be as specified in clause 8.4.3.6 of the present document.

7.2.4.3 Reporting results in JSON protocol

The result of the validation of a certain signature shall be reported in the `result` child element of the `resForOneSig` element reporting the processing results for that signature. The contents of this element shall be as specified in clause 8.4.3.1 of the present document.

The result of the attempt to augment a certain signature shall be reported in the `result` child element of the `augmentSigResult` element reporting the results of that augmentation attempt. The contents of this element shall be as specified in clause 8.4.3.6 of the present document.

8 Processing models

8.1 Introduction

The present clause specifies the processing models for the three protocols defined in the present document.

When the server receives a request from the client, it shall conduct the following tasks:

1) Retrieve the signature(s) that the client has requested to process. Clause 8.2 of the present document specifies models for this task.

2) Process all the signatures that the client requested to process. Clause 8.3 of the present document specifies models for this task.

3) Build the response for the client. Clause 8.4 of the present document specifies models for this task.

The models for retrieving the signatures to be processed shall depend on the type of signature to be retrieved. Models for retrieving XAdES and CAdES are based on the processing models for XML and CMS signatures specified in clauses 6.1 and 6.2 of DSS-X core v2.0 [1]. The model for retrieving PAdES signatures is fully specified in the present document as DSS-X core v2.0 [1] does not cover their submission and processing.

The models for retrieving the signatures to be processed shall be common to the three protocols, as the supporting structures for submitting signatures and signed documents are the same in all of them.

The models for processing the signatures shall depend on both, the types of signatures to be processed, and the protocol. In essence, the present document specifies two different types of signature processing, namely: validation and augmentation.

For servers implementing the 'validation' protocol the process applied to the signatures shall be their validation.

For servers implementing the 'augmentation' protocol the process applied to the signatures shall be their augmentation.

For servers implementing the 'validation and augmentation' protocol the process applied to the signatures shall be their validation and augmentation.

The model for building the responses shall be shared by all the protocols and all the types of signatures.

The processing models for validating XAdES and CAdES signatures are based on the processing models for XML and CMS signatures specified in clauses 6.1 and 6.2 of DSS-X core v2.0 [1]. Clauses below specify the changes in those processing models introduced by the particularities of the validation protocol specified in the present document.

The protocol specified in DSS-X core v2.0 [1] does not allow submission of PAdES signatures for their validation.

8.2 Retrieving signature(s)

8.2.1 Retrieving XAdES signature(s)

The process for retrieving the XAdES signatures in the request messages shall be as follows:

- If the request message does not include the optional input (specified in clause 5.1.4.2.1) for identifying the signatures to be processed, the server shall retrieve the XAdES signatures (built on XML signatures) as specified in clause 6.1.1 of DSS-X core v2.0 [1].

- If the request message includes the optional input for identifying the signatures to be validated, the server shall proceed as follows:
    - If this optional input contains the component enclosing digest values of the signatures to be processed, the server shall retrieve those signatures enclosed in (or referenced by) the signature object container, and it shall inspect all the input documents enclosed in (or referenced by) the different input documents containers searching for signatures embedded within them matching the aforementioned digest values.
    - If this optional input contains components enclosing a pointer to one XAdES signature, the server shall retrieve each XAdES signature pointed by each component, as specified in DSS-X core v2.0 [1].
- If the protocol implemented is the 'validation' or the 'validation and augmentation', then the server shall retrieve the documents that match the RefURI and RefType values, as specified in clause 6.1.2 of DSS-X core v2.0 [1].

8.2.2 Retrieving CAdES signature(s)

If the request message does not include the optional input (specified in clause 5.1.4.2.1) for identifying the signatures to be processed, the server shall retrieve the CMS structure as specified in step 1 of clause 6.2.1 of DSS-X core v2.0 [1].

The `signerInfos` set of this CMS structure may contain several items. Each item is considered to be part of one CAdES signature. Consequently, once retrieved the CMS structure, the server shall consider that it has as many CAdES signatures as items within the `signerInfos` set.

If the request message does include the optional input for identifying the signatures to be processed, the server shall retrieve those CAdES signatures enclosed in (or referenced by) the signature object container, that match the digest values present in the aforementioned optional input identifying the signatures to be processed.

8.2.3 Retrieving PAdES signature(s)

The process for retrieving the PAdES signatures in the request messages shall be as follows.

If the request message does not include the optional input (specified in clause 5.1.4.2.1) for identifying the signatures to be processed then:

- If the signature object container does enclose a pointer, the server shall retrieve the PDF document present or referenced within the document container referenced by that pointer and it shall then retrieve all the PAdES signatures embedded in that document.
- If the signature object container encloses the PAdES signature itself, the server shall retrieve the digest of the signed document from the digest document container.

If the request message includes the optional input (specified in clause 5.1.4.2.1) for identifying the signatures to be processed then:

1) The server hall retrieve the PDF document present or referenced within the document container referenced by the pointer within the signature object container and it shall then retrieve all the PAdES signatures embedded in that document.
2) The server shall then retain those PAdES signatures referenced in the optional input (specified in clause 5.1.4.2.1) for identifying the signatures to be processed then the server as follows:
    - If this optional input contains the component enclosing digest values of the signatures to be processed, the server shall retain those signatures that match the aforementioned digest values.
    - If this optional input contains the component identifying PDF field names, the server shall retain the signatures placed within the PDF fields whose names match any of the values present in the aforementioned optional input.

NOTE: At the end of this task, the server has retrieved either a non empty set of PAdES signatures and one PDF document, or one PAdES signature and one document digest.

8.2.4 Non retrieved signatures

In any of the three cases described in preceding clauses, for each signature reference that the server does not match with any signature in the request, the server shall generate one signature results container with only the result component, with its major result set to `urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError`, and its minor result set to `http://uri.etsi.org/19442/v1.1.1#SignatureNotLocated`.

8.3 Processing signature(s)

8.3.1 Validating signature(s)

8.3.1.1 Validating XAdES signature(s)

For each XAdES signature retrieved after conducting the task specified in clause 8.2.1 the server:

1) Shall conduct the sub-process 'recalculate references' as specified in clause 6.1.2 of DSS-X core v2.0 [1].

2) Shall conduct its validation as specified in ETSI TS 119 102-1 [13].

8.3.1.2 Validating CAdES signature(s)

The server shall first retrieve either the signed document as specified in step 2 of clause 6.2.1 of DSS-X core v2.0 [1], or retrieves the document digest from the digest document container.

For each CAdES signature retrieved after conducting the task specified in clause 8.2.2, the server shall conduct its validation as specified in ETSI TS 119 102-1 [13].

8.3.1.3 Validating PAdES signature(s)

For each PAdES signature retrieved after conducting the task specified in clause 8.2.3, the server shall conduct its validation as specified in ETSI TS 119 102-1 [13].

8.3.2 Augmenting signature(s)

For each AdES signature retrieved after conducting the task specified in the sub-clause of clause 8.2 corresponding to the AdES signature type, the server shall proceed to augment it to the level indicated in the component (specified in clause 6.1.2.2 of the present document) identifying the level that the signatures have to be augmented to.

For XAdES signatures the augmentation shall be conducted incorporating unsigned attributes/properties as specified in ETSI EN 319 132 [3], ETSI TS 101 903 [7] or ETSI TS 103 171 [8].

For CAdES signatures the augmentation shall be conducted incorporating unsigned attributes/properties as specified in ETSI EN 319 122 [2], ETSI TS 101 733 [5] or ETSI TS 103 173 [6].

For PAdES signatures the augmentation shall be conducted incorporating unsigned attributes/properties as specified in ETSI EN 319 142 [4], ETSI TS 102 778 [9] or ETSI TS 103 172 [10].

8.4 Building response message

8.4.1 Introduction

This task shall be affected by the optional inputs present in the request message.

Clause 8.4.3 specifies how to build the new optional outputs originated by the new optional inputs specified in clauses 5.1.4.2 and 6.1.2 of the present document.

Clause 8.4.5 specifies where the server shall place the different optional outputs, and what values shall the server assign to the different result components present within the response message.

8.4.2 Building the global result component

For the responses of any of the three protocols specified in the present document, if the server has not been able to process the request, it shall generate a result component with its result major set to a value indicating the cause of the failure, as specified in clause 4.1.7 of DSS-X core v2.0 [1]. Under these circumstances this component shall not have a result minor component.

NOTE 1: According to DSS-X core v2.0 [1], when the server is able to complete the processing of the signatures requested, i.e., when it has been able to retrieve and validate all of them, and regardless whether all of them are valid or not, the result major of the result component is `urn:oasis:names:tc:dss:1.0:resultmajor:Success`, and whose minor result may have different values depending on the results of the processing. Otherwise, the server does not process any signature and returns a result major different than the former one indicating the reasons preventing the processing, without any result minor.

If the server has been able to successfully process the request the following requirements apply.

NOTE 2: In the former sentence "the server has been able to successfully process the request" means that the request was fully correct, that the server correctly understood it, and that the server tried to process (validate, augment, or validate-and-augment depending on the protocol) all the signatures identified in the request. See the rest of this clause (for cases where the response does not contain any signature processing results container), and clauses 8.4.2 and 8.4.3.6 (for cases where the response contains one or more signature processing results containers) for more details on how to report on each process applied to each signature.

For the responses of the "validation" protocol that do not include any signature results container, the server shall generate the following contents:

NOTE 3: This only happens under certain circumstances, see clause 8.4.3.1 for details.

- If the server has been able to successfully process the request, then:
  - the result major shall have the value urn:oasis:names:tc:dss:1.0:resultmajor:Success AND
  - The result minor shall indicate the result of validating the signature, as follows:
    - If the validation has succeeded (total-passed as specified in ETSI TS 119 102-1 [13]), it shall have the following value: `http://uri.etsi.org/19442/validation/signature/totalpassed`.
    - If the validation returns the value indeterminate as specified in ETSI TS 119 102-1 [13], it shall have the following value: `http://uri.etsi.org/19442/validation/signature/indeterminated`.
    - If the validation returns the value total-failed as specified in ETSI TS 119 102-1 [13], it shall have the following value: `http://uri.etsi.org/19442/validation/signature/totalfailed`.
  - The result component may also provide textual information on the causes for this result.

For the responses of the "validation" protocol that include one or more signature results container, and for all the responses the "augmentation" and the "validation and augmentation" protocols the server shall generate a global result component as follows:

- It shall set its major result value to `urn:oasis:names:tc:dss:1.0:resultmajor:Success` indicating that it was able to complete the requested.
- It shall set the value of the result minor to `http://uri.etsi.org/19442/v1.1.1/result/CheckIndividualResults`.

8.4.3 Building new optional outputs

8.4.3.1 Building the signature processing results container

The server shall include one or more processing signature containers within the response message:

- if the message is the response for a request of the 'augmentation' OR 'validation and augmentation' protocols; OR
- if the messages is the response for a request of the 'validation protocol' AND:
    - The request message included the signatures-to-process-refs container (specified in clause 5.1.4.2.1 of the present document) identifying the signatures that the client requested to process; OR
    - The request message did NOT include signatures-to-process-refs container AND the request message:
        - contained several signatures; OR
        - contained one signature AND included the optional input (specified in clause 5.1.4.2.3 of the present document) requesting the generation of a (signed or unsigned) validation report.

Otherwise the server does not need to generate and include this component within the response message (although it still may do it).

NOTE: This allows that when operating in the validation protocol, if the request contains one signature and it does not contain any of the new optional inputs specified in the present document, the response may actually be the response message specified in DSS-X core v2.0 [1] with the only difference of the result minor value indicating the result of the validation.

The server shall generate one component of this type for each processed signature.

The server shall place all the components of this type as children of the optional outputs container (specified in clause 5.1.4.1 of the present document).

Each component shall have one child summarizing the result of the validation of one signature. The contents of its result major and result minor shall be as specified in DSS-X core v2.0 [1], with the following difference:

1) If the validation of the reported signature in this component has succeeded (total-passed as specified in ETSI TS 119 102-1 [13]), the result major shall have the following value: `http://uri.etsi.org/19442/validation/signature/totalpassed` and there shall not be any result minor.
2) If the validation of the reported signature in this component returns the value indeterminate as specified in ETSI TS 119 102-1 [13], the result major shall have the following value: `http://uri.etsi.org/19442/validation/signature/indeterminated` and there shall be a result minor component identifying one cause for this result. The result component may also provide textual information on the causes for this result.
3) If the validation of the reported signature in this component returns the value total-failed as specified in ETSI TS 119 102-1 [13], the result major shall have the following value: `http://uri.etsi.org/19442/validation/signature/totalfailed` and there shall be a result minor component indicating identifying the reason for this failure. The result component may also provide textual information on the causes for this result.

In a response message for the "validation and augmentation" protocol, this component may also have a augment results container child, whose building is specified in clause 8.4.3.6 of the present document.

8.4.3.2 Building component for referencing the processed signature

The server shall generate this component (specified in clause 5.2.3.2.2 of the present document) if it also generates one or more instances of the component for including details corresponding to the processing of one signature.

The server shall generate one component of this type for each processed signature, and shall place it as child component of the component that encloses the processing details for the referenced signature.

8.4.3.3 Building component for returning the (signed or unsigned) validation report

The server shall generate this component (specified in clause 5.2.3.2.5 of the present document) if the request included the optional input (specified in clause 5.1.4.2.3 of the present document) requesting the generation of a (signed or unsigned) validation report.

If the server has been requested to validate a set of N signatures, it may choose either:

1) generating one unique validation report reporting on the validation of the N validated signatures; in this case it shall incorporate this validation report component into the validation response message as a child of the optional outputs component; OR

2) generating as many validation reports (N) as validated signatures, each one reporting on the validation of one signature; in this case it shall incorporate each validation report as a child of the signature processing results container with the details of the validated signature; OR

3) implementing a mixed solution:

   - generating one validation report with details of the validation of a subset of M (being M<N) validated signatures; as in bullet 1) it shall incorporate it into the response message as child of the optional outputs component; AND

   - generating N-M validation reports each one reporting on the validation of one of the N-M signatures not present in the aforementioned subset; as in bullet 2), it shall incorporate each one of these N-M validation reports as child of the signature processing results container with the details of the validated signature.

8.4.3.4 Building component for notifying the signature validation policy applied during the validation

The server shall include this component (specified in clause 5.2.3.2.3 of the present document) if the request included the optional input (specified in clause 5.1.4.2.2 of the present document) requesting that the server uses a certain signature validation policy for validating any signature.

The server shall place this component as direct child of the optional outputs container component.

NOTE: Because the server uses the same signature validation policy for validating all the signatures, there is no need to repeat this information within each component notifying the validation details of one signature: instead, the component notifying the applied signature validation policy is placed as a sibling component.

8.4.3.5 Building component for returning the signature policies available

The server may include this component (specified in clause 5.2.3.2.4 of the present document) if the request included the optional input (specified in clause 5.1.4.2.2 of the present document) requesting that the server uses a certain signature validation policy for validating any signature AND the server could not validate signatures applying the requested signature validation policy.

The server shall place this component as direct child of the optional outputs container component.

NOTE: This component is placed there for the same reasons as the component for notifying the applied signature validation policy.

8.4.3.6 Building the augment signature result container

The server shall build this component (specified in clause 6.2.3 of the present document) only for the response messages of the 'augmentation' and the 'validation and augmentation' protocols.

The server shall build one component of this type for each signature whose augmentation has been requested by the client.

The server shall place this component as a child of the signature processing results container (specified in clause 5.2.3.2.1) in response messages of "validation and augmentation" protocol.

The server shall place this component as a child of the root element in response messages of "augmentation" protocol.

For each augmented signature the server shall generate this component as follows:

- If the signature was not embedded within any document then the server shall place the augmented signature within this component.
- If the signature was embedded within an embedding document, then the server:
    - Shall place the augmented signature within the embedding document, shall create a new document-with-signature container, and shall place the embedding document there.
    - Shall generate a pointer to the aforementioned document-with-signature container and shall put it within this component.

If the server has succeeded in the augmentation of the signature referenced in this component, this component:

1) Shall have the three elements enumerated in clause 6.2.3.1.
2) Shall have a result component with its major result child set to a value indicating that the process has succeeded and with its minor result child set to `http://uri.etsi.org/19442/v1.1.1/augmentation/Success`.

If the server has not succeeded in the augmentation of the signature this component:

1) Shall have a reference to the signature that the client requested to validate and augment in the augmentation request.
2) Shall not contain the element allowing to return the augmented signature.
3) Shall have a result component with its major result child set to `http://uri.etsi.org/19442/v1.1.1/augmentation/Failure`. The result component may have a minor result child. Below follows a list of possible causes for augmentation failure:
    - If the client requested augmenting the signature to a level that the signature cannot be augmented to, then the minor result child shall be set to `http://uri.etsi.org/19442/v1.1.1/augmentation/Forbidden`.

EXAMPLE: Requesting augmentation to AdES-B-B level is never possible. Also, if a signature is a XAdES-B-L-T, a client can only request augmentation to AdES-L-TA level. Any other augmentation request will fail with the result minor set to `http://uri.etsi.org/19442/v1.1.1/augmentation/Forbidden`.

- If the server does not know the signature level indicated in the request then the minor result shall be set to `http://uri.etsi.org/19442/v1.1.1/augmentation/unknownLevel`.
- If the server is not prepared for augmenting the signature to the level indicated in the request then the minor result shall be set to `http://uri.etsi.org/19442/v1.1.1/augmentation/notPreparedForThisLevel`.
- If the server could not get all the material required for augmenting the signature then the minor result child shall be set to `http://uri.etsi.org/19442/v1.1.1/augmentation/validationMaterialNotAvailable`.
- Any other cause shall be reported setting the minor result child to `http://uri.etsi.org/19442/v1.1.1/augmentation/otherCause`.

8.4.4 Building DSS-X re-used optional outputs

The building of the DSSX re-used optional outputs present in the response message shall be performed as specified in clause 6.3.3 of DSS-X core v2.0 [1].

8.4.5 Building the response message

If the server has found itself in the situation of not being able to complete the request, it shall generate a response message with only one global result component. Its result major shall identify the reason that prevented the server to successfully process the request, as specified in clause 4.1.7 of DSS-X core v2.0 [1]. This result component shall not contain any result minor.

NOTE 1: According to DSS-X core v2.0 [1], when the server is able to complete the processing of the signatures requested, i.e., when it has been able to retrieve and validate all of them, and regardless whether all of them are valid or not, the result major of the result component is `urn:oasis:names:tc:dss:1.0:resultmajor:Success`, and whose minor result may have different values depending on the results of the processing. Otherwise, the server does not process any signature and returns a result major different than the former one indicating the reasons preventing the processing, without any result minor.

If the server has been able to process all the signatures whose processing the client requested:

1) The server shall generate a global result component as follows:

   a) It shall set its major result value to `urn:oasis:names:tc:dss:1.0:resultmajor:Success` indicating that it was able to complete the requested.

   b) It shall set the value of the result minor as follows:

      ▪ If the response does not include any signature processing results container (specified in clause 5.2.3.2.1 of the present document), meaning that only one signature has been validated, then value for the result minor shall be one of the three values specified in clause 8.4.3.1 of the present document, depending on the result obtained by the validation process.

NOTE 2: This can only happen in the response of the 'validation' protocol.

      ▪ If the response does include one or more signature processing results containers, then the value for the result minor shall be `http://uri.etsi.org/19442/v1.1.1/result/CheckIndividualResults`.

2) If the response does NOT include any signature processing results container, then the server proceeds to generate the optional outputs as per the optional inputs present in the request, following the specifications in clause 6.3.3 of DSS-X core v2.0 [1], and clauses 8.4.3.4, and 8.4.3.5 of the present document. After that, it places the generated optional outputs as children of the optional outputs container component.

NOTE 3: This can only happen in the response of the 'validation' protocol.

3) For responses of the 'validation' and 'validation and augmentation' protocols, if the response does include one or more signature processing results containers, the server:

   - Shall build all the optional outputs common to all the processed signatures as per the optional inputs present in the request, namely the components containing: the applied service policy, the schemas used, the documents embedding signatures, the verification time information, and the notification of the applied signature validation policy (or the available signature validation policies). The server shall place all these components as children of the optional outputs container. The server shall build all these components as specified in DSS-X core v2.0 [1] and the present document.

   - Shall build one signature processing results container per each processed signature. For each processed signature, the server proceeds to generate the components reporting one aspect of its processing as per the optional inputs in the request, and places them as children of the signature processing results container. These children components are: a signature reference component, a result component reporting the result of processing that signature, the (signed or unsigned) validation report, the augment signature result container (which shall be built as indicated in the next bullet), transformed documents, results of verifying manifests, the signing time information, and the signer identity.

4) For responses of the 'augmentation' and the 'validation and augmentation' protocols:

   - The server shall build one augment signature result container specified in clause 7.2.3.2 of the present document for each processed signature.

- The server shall set the value of the result component in each augment signature result container as specified in clause 6.2.3.1 of the present document.

5) For the 'augmentation' protocol the server shall place the augment signature results containers as children of the root element of the response.

6) For the 'augmentation' protocol, the server shall place the document-with-signature containers generated during the processing of the signatures as children of the root element of the response. For the 'augmentation and validation' protocol, the server shall place the document-with-signature containers generated during the processing of the signatures as children of the optional outputs component.

7) Shall set the values of the profiles used. The server shall set the value of the first one as specified in the present document, depending on the protocol.

8) May set values for the components containing the request identifier, and the response identifier.

9 Asynchronous processing

9.1 Asynchronous operation for the three protocols

For all the protocols specified in the present document, the server shall not send back to the client a response message that contains the results of processing only a subset of the set of signatures that the client requested to process.

The asynchronous processing allows to the server sending back to the client "not yet finished" messages, which do not contain any result of processing any signature, but serve to notify the client that it has to wait for getting the results it asked for.

In asynchronous processing one client usually sends an initial request to the server. The initial request shall contain a component with the request identifier generated by the client.

As a response to that initial request, if after a certain time the server has not been able to finalize the processing of all the signatures (either their validation, their augmentation, or their validation-and-augmentation) that the client requested to process, it shall return a protocol response with a component set to a value that indicates that it has not been able to finalize the processing of all the signatures ("not yet finished"). This response-pending message shall contain an identifier. This response-pending message shall not contain any result of processing any signature.

Under this processing mode the client, after a certain time, can send a pending-request to the server. This pending-request shall include the response identifier of the response-pending message returned by the server. This response identifier allows the server to correlate this pending-request to the initial request. If the server has finalized the processing of all the signatures that the client requested to process, it shall return the response message with the corresponding results; otherwise, it shall return a new response-pending.

The client can send subsequent requests until the server returns a response with the result of processing all the signatures that the client requested to process. Each subsequent request shall include the response identifier returned by the server in the response to the initial request.

For managing asynchronous processing, the following components are required in the three protocols:

a) An identifier for the initial request.

b) An identifier for the first response-pending message, which shall be used in subsequent pending-request messages.

c) A pending-request message that includes the identifier for the first response-pending message.

9.2 Components for sending a pending-request

9.2.1 Components semantics

This component shall request to the server to return the response corresponding to an initial request previously sent. Requests of this type are named pending-request hereinafter.

This component shall include an identifier, generated by the server and returned by the server in the response to the initial request (the first response-pending message), which allows correlating the subsequent pending requests to the initial request and to the subsequent responses.

9.2.2 XML component

The element that shall indicate to the server that the client is requesting the response corresponding to a previously sent initial request (as part of an asynchronous protocol) shall be the `dss2:PendingRequest` element specified in clause 4.2.12.2 of DSS-X core v2.0 [1].

9.2.3 JSON component

The element that shall indicate to the server that the client is requesting the response corresponding to a previously sent initial request (as part of an asynchronous protocol) shall be the `pendingReq` element. It shall be an instance of the `dss2-PendingRequestType` type specified in clause 4.2.12.1 of DSS-X core v2.0 [1].

9.3 Component for identifying the initial request

9.3.1 Component semantics

This component shall contain a unique identifier for the initial request submitted by the client to the server.

9.3.2 XML component

The component implementing the identifier of the initial request shall be the `RequestID` attribute specified in clause 4.1.10.2 of DSS-X core v2.0 [1] for the "validation" protocol and the "validation and augmentation" protocol.

The component implementing the identifier of the initial request shall be the `RequestID` attribute specified in clause 6.1.1.2 of the present document for the "augmentation" protocol.

9.3.3 JSON component

The component implementing the identifier of the initial request shall be the `reqID` component specified in clause 4.1.10.1 of DSS-X core v2.0 [1] for the "validation" protocol and the "validation and augmentation" protocol.

The component implementing the identifier of the initial request shall be the `reqID` component specified in clause 6.1.1.3 of the present document for the "augmentation" protocol.

9.4 Component for identifying the initial response-pending message

9.4.1 Component semantics

This component shall contain a unique identifier for the initial response-pending message sent by the sender to the client.

9.4.2 XML components

The component implementing the identifier of the initial request shall be the `ResponseID` attribute specified in clause 4.1.11.2 of DSS-X core v2.0 [1] for the "validation" protocol and the "validation and augmentation" protocol.

The component implementing the identifier of the initial request shall be the `ResponseID` attribute specified in clause 6.2.3.2 of the present document for the "augmentation" protocol.

9.4.3 JSON component

The component implementing the identifier of the initial request shall be the `respID` component specified in clause 4.1.11.2 of DSS-X core v2.0 [1] for the "validation" protocol and the "validation and augmentation" protocol.

The component implementing the identifier of the initial request shall be the `respID` component specified in clause 6.2.3.3 of the present document for the "augmentation" protocol.

Annex A (normative): XML and JSON Schema files

A.1 XML Schema file location for namespace http://uri.etsi.org/19442/v1.1.1#

The file at https://forge.etsi.org/rep/esi/x19_442_sign_validation_protocol/raw/v1.1.1/19442xmlSchema.xsd ("19442xmlSchema.xsd") contains the definitions of elements and types defined within the namespace whose URI value is http://uri.etsi.org/19442/v1.1.1#.

A.2 JSON Schema file location for "$schema" "http://etsi.org/19442/v1.1.1/json#"

The file at https://forge.etsi.org/rep/esi/x19_442_sign_validation_protocol/raw/v1.1.1/19442jsonSchema.json ("19442jsonSchema.json") contains the definitions of elements and types defined within the JSON schema associated to the present document.

Annex B (informative): Bibliography

- IETF RFC 7515: "JSON Web Signature (JWS)".
- IETF RFC 5652 (September 2009): "Cryptographic Message Syntax (CMS)".
- ETSI TS 119 172-4: "Electronic Signatures and Infrastructures (ESI); Signature policies; Part 4: Signature validation policy for European qualified electronic signatures/seals using trusted lists".

History

Document history		
V1.1.1	February 2019	Publication