DELIVERABLE

Project Acronym: **PEPPOL**

Grant Agreement number: **224974**

Project Title: **Pan-European Public Procurement Online**

PEPPOL Deliverable D1.3 Demonstrator and functional Specifications for Cross-Border Use of eSignatures in Public Procurement

Part 6: OASIS DSS Interface Specification

Extended Validation Profile – Profiling and Extensions Specification

Revision: 2.1

Authors:

Germany: bremen online services

Norway: Difi

Italy: InfoCamere, InfoCert

France: ADETEF, DILA, Lex Persona, ANSSI, Esteral Consulting

Greece: University of Piraeus

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	X
C	Confidential, only for members of the consortium and the Commission Services	

Revision History

Revision	Date	Author	Organisation	Description
1.0	2009/02/11			Complete version of D1.1 for internal quality assurance.
1.1	2009/02/27			D1.1 submitted to PEPPOL project management, approved with comments at project management meeting 2009/03/27.
1.2	2009/04/30			D1.1 for publication, updated according to comments.
1.3	2009/11/06			Formal update of D1.1 after EC approval.
1.8	2010/09/22			Complete D1.3 version edited from D1.1 part 6. For internal quality assurance.
1.9	2010/09/30			D1.3 submitted to PEPPOL project operating office (POO) for approval.
1.9.5	2010/11/05			D1.3 ready for publication, updated according to comments from POO. Uploaded for EC approval.
2.0	2010/07/15			Formal update after EC approval.
2.1	2011/08/30			Implementation of EC recommendations.

Statement of originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Statement of copyright

Table of Contents

1 Summary and Structure of Document

1.1 Scope and Structure of Deliverable D1.3

This document is a part of the multi-part deliverable D1.3 "Functional Specifications for Cross-Border Use of eSignatures in Public Procurement" issued by the PEPPOL[1] (Pan-European Public Procurement On-Line) project. PEPPOL is a 4-year (May 2008 – end April 2012[2]) large scale pilot under the CIP (Competitiveness and Innovation Programme) initiative of the European Commission. D1.3 is an updated version of the deliverable D1.1 "Requirements for Use of Signatures in the Procurement Processes" [PEPPOL-D1.1].

D1.3 consists of the following documents:

Part 1: Background and Scope

(Part 2: Not included – was the D1.1 part on E-tendering Pilot Specifications)

Part 3: Signature Policies

Part 4: Architecture and Trust Models

Part 5: XKMS v2 Interface Specification

Part 6: OASIS DSS Interface Specification

Part 7: eID and eSignature Quality Classification

The D1.3 deliverable is the second version of **functional specifications** for cross-border interoperability of e-signatures in Europe. The specifications are specifically targeted at cross-border public procurement, the topic of PEPPOL. However, a successful solution should be applicable also to other application areas in need of e-signature interoperability.

Signature interoperability in PEPPOL focuses on verification of e-signatures and their associated eIDs. Interoperability of signing solutions is not handled as it is assumed that all actors are capable of signing documents within their corporate infrastructure.

The specifications in deliverable D1.1 has guided the implementation and testing of e-signature interoperability solutions in PEPPOL. In the course of this work, the specifications have by necessity evolved, leading to the revised version published in this deliverable D1.3. These are the specifications for the solutions used for the e-signature interoperability pilots in PEPPOL [PEPPOL-D1.2] in the period 1st November 2010 to 30th April 2012.

The specifications are publicly available and comments from any interested party are most welcome. Note that further evaluation of the specifications of D1.3 is expected as a result of further work in PEPPOL and any party using or referring to the specifications must ensure that the latest version is used; contact the PEPPOL project for information.

1.2 Demonstrator Software Components and Documentation

In addition to the specifications in this deliverable D1.3, PEPPOL WP1 provides software components for cross-border validation of e-signatures:

[1] http://www.peppol.eu

[2] Originally, PEPPOL was scheduled for 3 years. The project has been prolonged twice, both times by 6 months.

- PEPPOL XKMS responder component (server side component) according to the specifications of D1.3 part 5 is provided as open source. The software component, source code and documentation are available on OSOR[3],
- A free to use client side component for signature validation can be retrieved from bremen online services. The validation client is available as a standalone version and a version for integration into other software applications. To receive download permission, please use the following contact:

bremen online services Support and supply of PEPPOL WP1 software components
• Phone: +49-421-20495-777 • E-Mail: support-wp1@peppol.eu

The software components are used for PEPPOL's pilot demonstrators on e-signature interoperability as described in PEPPOL Deliverable D1.2 [PEPPOL-D1.2]. Attachments A and B to D1.2 provide documentation on respectively the XKMS responder and the validation client.

1.3 Scope and Structure of this Document

Cross-border interoperability for verification of e-signatures requires more information than merely an assessment that the signature is valid. Signature validity is just one aspect of signature acceptance, which is governed by the signature policy in force (see D1.3 part 3).

PEPPOL specifies validation services and their interfaces. A validation service must be able to assess and return information related to signature policy adherence, which necessitates a richer interface than merely OCSP or CRL for revocation checking. Two interfaces are specified:

- XKMS v2 for eID certificate validation (part 5 of D1.3);
- OASIS DSS for verification of entire signed documents (this document).

The interface specified by this document provides an opportunity for external verification of entire signed documents (all signatures on a document). A verification service may be a technical service providing functionality only, or it may be an *authority* issuing signed responses that can be viewed as "notary" statements on validity of signatures.

Below, an overview of the specification is given in chapter 2 and identifiers and notational conventions are given in chapter 3. Request and response elements are specified in chapters 4 and 5 respectively. Profile bindings are given in chapter 6. The XML schema is provided in the Appendix.

1.4 Evolution of this Document and Changes from D1.1

Note: This document, like the other parts of D1.3, continues the version numbers deriving from D1.1.

[3] Open Source Observatory and Repository for European public administrations, http://www.osor.eu. Results from PEPPOL are available in http://www.osor.eu/projects/peppol.

Since the publication of PEPPOL Deliverable D1.1 end April 2009, no actor has committed to provision of an OASIS DSS interface for PEPPOL pilots. Thus, to prioritise resource usage, no further work has been done on this interface specification.

The only material changes since D1.1 are addition of references to the ongoing OASIS DSS-X work and some updates resulting from changes to other parts of D1.3 from the D1.1 version (e.g. namespaces as defined in both this document and the XKMS part, D1.3 part 5, have changed).

In addition, some restructuring of the document has been done, notably dividing the content into several chapters with fewer levels of section headers.

Although the updates are minor, a decision has been done to re-issue the specification as part of D1.3 since the information is believed to be useful.

If work on the OASIS DSS interface is continued, the specification should be more closely aligned with the XKMS specification, e.g.:

- Structure and semantics of statements about eIDs should be more closely aligned between the two specifications.
- The alignment may be to the extent that the XKMS structure is embedded into the DSS structure as the certificate validation part, DSS adding only the signature specific items and the overall (aggregated) status information.
- The DSS part should incorporate return of accreditation/supervision and quality (qualified with or without SSCD – Secure Signature Creation Device) from the EU's TSL (Trust Status List) system in addition to the quality classification system defined by D1.3 part 7. See D1.3 parts 4 and 5 for information.
- At present, a difference between the XKMS and DSS profiles is that XKMS assumes that the caller is able to process certificate content while for DSS certificate content may be returned as "respond with" parameters assuming that the caller does not need to do this processing.

If (or when) work on the OASIS DSS profile specified in this document is continued, this should be in close co-operation with the OASIS DSS-X (DSS eXtended) technical committee[4].

- The specification should be aligned with OASIS DSS-X work [DSS-MultiSign] or alternatively be promoted to a separate OASIS DSS standards profile.
- No comparison between the specifications in this document and the OASIS DSS-X specifications has been done, thus the effort needed for alignment has not been estimated.

1.5 List of Contributors

The following organisations, in alphabetical order, have contributed to Deliverable D1.3:

- **ADETEF, France** http://www.adetef.fr
- **ANSSI, French Network and Information Security Agency, France** http://www.ssi.gouv.fr
- **bos, bremen online services, Germany,** http://www.bos-bremen.de
- **Difi, Agency for Public Management and eGovernment, Norway** http://www.difi.no
- **DILA, Direction de l'Administration Légale et Administrative Of French Prime Minister Office, France** http://www.dila.premier-ministre.gouv.fr
- **Esteral Consulting, France** http://www.esteralconsulting.com

[4] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dss-x

- **InfoCamere, Italy** http://www.infocamere.it
- **InfoCert, Italy** http://www.infocert.it
- **Lex Persona, France** http://www.lex-persona.com
- **University of Piraeus, Greece** http://www.unipi.gr

The following persons (alphabetical ordering for each participating organisation) have contributed to the D1.3 work:

Jörg Apitzsch	bos	Piero Milani	InfoCamere	Alain Ducass	ADETEF
Nils Büngener	bos	Luca Boldrin	InfoCert	Ahmed Yacine	DILA
Mark Horstmann	bos	Daniele Mongiello	InfoCert	François Devoret	Lex Persona
Ralf Lindemann	bos	Lefteris Leontaridis	Univ. Piraeus	Julien Pasquier	Lex Persona
Dr Jan Pelz	bos	Dr Andriana Prentza	Univ. Piraeus	Sébastien Herniote	ANSSI
Lars Thölken	bos	Alain Esterle	Esteral Cons.	Jon Ølnes (editor)	Difi

D1.3 is a revised version of D1.1. The D1.3 team acknowledges the contributions of organisations and persons that helped producing D1.1 but are no longer active in PEPPOL's e-signature work. These are not listed above; please refer to D1.1 for the names.

2 Overview

This document defines a profile of the verifying protocol defined in “Digital Signature Services Core Protocol and Elements” [DSSCore]. The profile “Extended Validation profile” extends the verifying protocol with quality assessments and information regarding the signatures, the belonging certificates and the verification process.

The profile supports multi signature documents. However it only supports one document per request. The signature format used is not considered or limited by this profile; hence it should be possible to support any signature format. Further development of the specifications in this document should as far as possible be aligned with OASIS DSS-X work on multi-signature verification [DSS-MultiSign]. No analysis has been done on the differences between the DSS-X work and the specifications in this document, the feasibility of aligning the specifications or the effort needed.

The quality assessment for eID quality and signature quality is based on the quality assessment and classification scheme described in part 7 of the D1.3 deliverable. This should be enhanced to additionally incorporate return of accreditation/supervision and quality (qualified with or without SSCD – Secure Signature Creation Device) from the EU’s TSL (Trust Status List) system. See D1.3 parts 4 and 5 for information

Information about the signature and the belonging certificates is given in a new Optional Input / Optional Output. This is information retrieved from the different fields in an X509 certificate(s) and from the signature(s). For each signature verification, the status of that particular verification is reported in addition to an overall status for all signature verifications.

An OASIS DSS validation service as specified in this document shall be able to assess aspects of signature policy adherence, notably that quality requirements are fulfilled. If comprehensive and unambiguously identified signature policies are defined (a more pragmatic approach is taken at present by D1.3 part 3), identification of signature policy and “verified according to policy” indication can be signaled over the DSS interface, e.g. according to [DSS-SigPolicy].

The time of verification of the signatures and forming of the response are set as required in this profile.

The profile opens up for asynchronous communication. Communication through a validation gateway is also possible. A validation gateway may be beneficial e.g. when it is desirable to have all requests from a company towards a validation service, from all systems inside a company, go through a common channel towards the validation service. The validation gateway can then sign all requests on behalf of the company and common policies, e.g. quality levels, can be enforced by the validation gateway. The validation gateway can also remove document content to avoid sending the content to the validation service (for confidentiality reasons or simply to save bandwidth), forwarding only signature fields and corresponding hash values. The concept of a validation gateway is briefly discussed in D1.3 part 4.

3 Profile Identification, Scope, Notational Conventions

3.1 Identifier

urn:oasis:names:tc:dss:1.0:profiled:extended-validation

3.2 Scope

This document profiles the DSS verifying protocol defined in [DSSCore]. This profile extends the core specification with quality assessment and more information in the response.

3.3 Relationship to other Profiles

The profile in this document is based on the [DSSCore]. It MAY be used with other profiles of the [DSSCore].

3.4 Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]7.

This profile uses the following typographical conventions in text:

<Element>, **<ns:ForeignElement>**, **Attribute**, **Type**, **OtherCode**

```
Listings of Extended Validation profile schema like this.
```

3.5 Namespaces

Prefix	XML Namespace	Specification
ds	**http://www.w3.org/2000/09/xmldsig#**	[XMLDSIG]
dss	**http://docs.oasis-open.org/dss/v1.0/oasis-dss-core-schema-v1.0-os.xsd** **urn:oasis:names:tc:dss:1.0:core:schema**	[DSSCore]
evdss	**http://www.peppol.eu/....**	This document
xkms	**http://www.w3.org/2002/03/xkms#**	[XKMS]
xkmsEU	**http://uri.peppol.eu/xkmsExt/v2#**	D1.3 part 5
xs	**http://www.w3.org/2001/XMLSchema**	[XMLSchema]

Table 1: Namespaces

4 Element <VerifyRequest>

This chapter profiles the **<VerifyRequest>** element.

4.1 Attribute RequestID

This attribute is REQUIRED.

The attribute **RequestID** SHOULD be a globally unique ID.

One solution for the content of the **RequestID** is to combine the **<Name>** element from **<RequesterIdentity>** with a 80-bit random number to make the ID approximately unique.

4.2 Attribute Profile

The attribute is REQUIRED.

urn:oasis:names:tc:dss:1.0:profiles:extended-validation

4.3 Element <OptionalInputs>

Except where otherwise stated, the elements in this section are new optional inputs defined by this document.

4.3.1 Element <QualityLevelRequirements>

This element is REQUIRED.

In cases where there are no requirements regarding quality, the requested quality SHALL be set to 0.

The element **<QualityLevelRequirements>** is based on the type **QualityLevelType**. Information about the quality assessment and a complete description is found in part 7 of the D1.1 deliverable, and enumerations are defined in part 5 of D1.1.

The **QualityLevelType** has the following child elements:

<CertificateQuality> [REQUIRED]

This element contains the minimum required certificate quality level.

<IndependentAssurance> [REQUIRED]

This element is the minimum required independent assurance level.

<HashAlgQuality> [REQUIRED]

This element contains the minimum accepted hash algorithm quality level.

<PublicKeyAlgQuality> [REQUIRED]

This element contains the minimum accepted public key algorithm quality level.

```
<!-- QualityLevelRequirements -->
<xs:element name="QualityLevelRequirements" type="QualityLevelType"/>
<xs:complexType name="QualityLevelType">
 <xs:sequence>
  <xs:element name="CertificateQuality" type="xs:string"/>
  <xs:element name="IndependentAssurance" type="xs:string"/>
  <xs:element name="HashAlgQuality" type="xs:string"/>
```

```
  <xs:element name="PublicKeyAlgQuality" type="xs:string"/>
 </xs:sequence>
</xs:complexType>
```

4.3.2 Element <RespondWith>

The **<RespondWith>** element is REQUIRED. The values **Subject**, **IssuerName** and **CertificateSerialNumber** are the minimum of the **RespondWithEnum** values that MUST be used.

```
<!-- RespondWith -->
<xs:element name="RespondWith" type="RespondWithEnum"/>
<xs:simpleType name="RespondWithEnum">
 <xs:restriction base="xs:string">
  <xs:enumeration value="Subject" />
  <xs:enumeration value="IssuerName"/>
  <xs:enumeration value="CertificateSerialNumber"/>
  <xs:enumeration value="KeyValue"/>
  <xs:enumeration value="HashAlgorithm"/>
  <xs:enumeration value="X509CertificateChain"/>
  <xs:enumeration value="X509CRL"/>
  <xs:enumeration value="SKI"/>
  <xs:enumeration value="OCSP"/>
  <xs:enumeration value="Timestamp"/>
  <xs:enumeration value="SignHash"/>
  <xs:enumeration value="ContentHash"/>
  <xs:enumeration value="Content"/>
  <xs:enumeration value="KeyUsage"/>
  <xs:enumeration value="ExtendedKeyUsage"/>
  <xs:enumeration value="BasicConstraints"/>
  <xs:enumeration value="ValidFrom"/>
  <xs:enumeration value="ValidTo"/>
  <xs:enumeration value="CRLUrl"/>
  <xs:enumeration value="CRLNumber"/>
 </xs:restriction>
</xs:simpleType>
```

The meanings of the different **RespondWithEnum** values are listed in the table below.

RespondWithEnum	Required / Optional	Meaning
Subject	REQUIRED	This is the distinguished name (DN) of the holder of the certificate (subject field in a certificate)
IssuerName	REQUIRED	This is the DN of the certificate issuer
CertificateSerialNumber	REQUIRED	This is the serial number in the certificate
KeyValue	OPTIONAL	This is the certificate holder's public key
HashAlgorithm	OPTIONAL	For a signature verification, this hash algorithm is extracted from the signature, while in a certificate validation this hash algorithm is found in the signature algorithm field in the certificate

RespondWithEnum	Required / Optional	Meaning
X509CertificateChain	OPTIONAL	This is the certificate chain for the certificate
SKI	OPTIONAL	The SKI (Subject Key Identifier) is the SKI from the certificate. This is an extension in the certificate, and it provides a means of identifying certificates that contain a particular public key
KeyUsage	OPTIONAL	This is the key usage from the certificate
ExtendedKeyUsage	OPTIONAL	This is the extended key usage from the certificate
BasicConstraints	OPTIONAL	This is the basic constraints for the certificate.
ValidFrom	OPTIONAL	The certificate is valid from this date
ValidTo	OPTIONAL	The certificate is valid to this date
SignHash	OPTIONAL	This is the decrypted hash of the signature
ContentHash	OPTIONAL	This is the hash of the content sent in the request
Content	OPTIONAL	This is the content sent in the request
X509CRL	OPTIONAL	This is the CRL used to validate the end user's certificate. If other methods than CRLs are used for validation of certificates, this value will be N/A. Note: The return of CRLs may reduce the response time of a request due to the size of the CRL
OCSP	OPTIONAL	If OCSP is used in the validation of a certificate, the entire OCSP response is given here. If OCSP is not used, this value will be N/A
CRLUrl	OPTIONAL	This is the URL from which the CRL was downloaded
CRLNumber	OPTIONAL	This is the CRLNumber of the CRL used for validation of the certificate
Timestamp	OPTIONAL	This is the timestamp of a signature. If a time stamp is not used, the value will be N/A

4.3.3 Element <GatewayRequester>

To support the use of a validation gateway forwarding requests to a validation authority (VA) service the element **<GatewayRequester>** has been defined. The element uses the **IdentityType** type which is based on the **<RequesterIdentity>** element. The **<GatewayRequester>** element is OPTIONAL, but it is REQUIRED when a validation gateway is being used to forward a request to a VA service.

<Name> [REQUIRED]

This is the name of the validation gateway the verification request is sent through. This element is of the type **saml:NameIdentifierType**. This SHOULD be filled with the CN from the validation gateway's client authentication certificate used in the two-way SSL connection.

<SupportingInfo> [OPTIONAL]

This element MAY include an URI to the requester or responder to support an asynchronous version of this protocol.

```
<!-- GatewayRequesterIdentity -->
<xs:element name="GatewayRequesterIdentity" type="IdentityType"/>
<xs:complexType name="IdentityType">
 <xs:sequence>
  <xs:element name="Name" type="saml:NameIdentifierType"/>
  <xs:element name="SupportingInfo" type="dss:AnyType" minOccurs="0"/>
 </xs:sequence>
</xs:complexType>
```

4.3.4 Element <UseVerificationTime>

The element **<UseVerificationTime>** is already defined in [DSSCore].

The element consists of two child elements; **<CurrentTime>** and **<SpecificTime>**. All elements are OPTIONAL. When the **<UseVerificationTime>** element is not used it is up to the server's policy to determine what time to use, e.g. current time or the time stamp in the SDO.

4.3.5 Element <ReturnVerificationTimeInfo>

The element **<ReturnVerificationTimeInfo>** is already defined in [DSSCore]. This element is made REQUIRED in this profile.

4.3.6 Element <RequesterIdentity>

The element **<RequesterIdentity>** is already defined in the [DSSCore]. This element is made REQUIRED in this profile.

The element is used to identify the requester of the verification. The child elements are as follow:

<Name> [REQUIRED]

This is the name of the requester who requested the verification. This element SHOULD be the same as the CN in the client authentication certificate used in the security bindings.

<SupportingInfo> [OPTIONAL]

This element may be used in an asynchronous communication between the client and the server. This element will then contain a respond address element

4.4 Element <InputDocuments>

This profile supports only one document in one request, but the document may have an unlimited number of signatures. The document is added using an appropriate child element of the **<InputDocuments>** element following the definitions in [DSSCore]. This element is REQUIRED in this profile. Instead of sending the entire document, a document hash and the signatures may be used.

4.5 Element <SignatureObject>

The **<SignatureObject>** element is only used in relation with the **<DocumentHash>** element in **<InputDocuments>**.

5 Element <VerifyResponse>

5.1 Attribute RequestID

This attribute is REQUIRED in this profile. The **`RequestID`** must be the same as the **`RequestID`** in the request.

5.2 Attribute Profile

The element is REQUIRED in this profile.

5.3 Element <Result>

5.3.1 Element <ResultMajor>

The values defined for **`<ResultMajor>`** in the core specification are used as specified there.

5.3.2 Element <ResultMinor>

It is not defined any new **`<ResultMinor>`** codes for this profile at the moment. Refinements of some of the codes are given below.

For the **`<ResultMajor>`** code **`Success`** the following **`<ResultMinor>`** codes are specified with example usage:

- **`IncorrectSignature`**: This code should be used when one of the signatures fails to verify and the **`OverallAssertionStatus`** attribute in the **`<ContentVerifyInfo>`** element is **`NotTrusted`**.
- **`ValidMultiSignatures`**: This code should be used when the signature verification are valid for all signatures

For the **`<ResultMajor>`** code **`InsufficientInformation`** the following **`<ResultMinor>`** codes are specified with example usage:

- **`CrlNotAvailable`**: This code is used if the CRL is unavailable during verification of any of the signatures resulting in an **`OverallAssertionStatus`** attribute in the **`<ContentVerifyElement>`** set to Indeterminate
- **`OcspNotAvailable`**: This code is used if the OCSP responder is unavailable during verification of any of the signatures resulting in an **`OverallAssertionStatus`** attribute in the **`<ContentVerifyElement>`** set to **`Indeterminate`**

5.4 Element <OptionalOutputs>

Except where otherwise stated, the elements in this section are new optional inputs defined by this document.

5.4.1 Element <ResponderIdentity>

The element **`<ResponderIdentity>`** is required in a response from a verification service using this profile. The element is of type **`IdentityType`**.

<Name> [REQUIRED]

This element SHOULD be populated with the CN of the service's signing certificate or the SSL Server authentication certificate used in the SSL connection.

<SupportingInfo> [OPTIONAL]

Not in use in this profile.

```
<!-- Responderidentity -->
<xs:element name="ResponderIdentity" type="IdentityType"/>
<xs:complexType name="IdentityType">
 <xs:sequence>
  <xs:element name="Name" type="saml:NameIdentifierType"/>
  <xs:element name="SupportingInfo" type="dss:AnyType" minOccurs="0"/>
 </xs:sequence>
</xs:complexType>
```

5.4.2 Element <QualityLevel>

The different quality levels SHALL always be returned when using the extended validation profile of DSS, hence the **<QualityLevel>** element is REQUIRED. The **<QualityLevel>** is of type **QualityLevelType**. The **<QualityLevel>** element is used in the **<VerifyInfo>** element described in 5.4.5. The **QualityLevel** consist of the following:

<CertificateQuality> [REQUIRED]

This is the actual quality of the certificate(s) used for signing.

<IndependentAssurance> [REQUIRED]

This is the independent assurance level of the certificate(s) used for signing.

<HashAlgQuality> [REQUIRED]

This is the quality of the hash algorithm used

<PublicKeyAlgQuality> [REQUIRED]

This is the quality of the public key algorithm used

```
<!-- QualityLevel -->
<xs:element name="QualityLevel" type="QualityLevelType"/>
<xs:complexType name="QualityLevelType">
 <xs:sequence>
  <xs:element name="CertificateQuality" type="xs:string"/>
  <xs:element name="IndependentAssurance" type="xs:string"/>
  <xs:element name="HashAlgQuality" type="xs:string"/>
  <xs:element name="PublicKeyAlgQuality" type="xs:string"/>
 </xs:sequence>
</xs:complexType>
```

5.4.3 Element <ResponseItem>

The **<ResponseItem>** element is of type **ResponseItemType**, and it is REQUIRED in this profile. This Optional Output is used in the **<VerifyInfo>** element. This element returns all the requested **RespondWithEnum** parameters. The **<ResponseItem>** element consists of the following elements and attribute:

<IntValue> [OPTIONAL]

The **<IntValue>** element is of type **integer** defined in [XMLSchema]. This element may be used for any **RespondWithEnum** item that has an integer value as result.

<Value> [OPTIONAL]

The **<Value>** element is of type **string** defined in [XMLSchema]. This element is for future use, and it can have multiple values against one **RespondWithEnum** item.

<Base64Value> [OPTIONAL]

The **<Base64Value>** element is of type **base64Binary** defined in [XMLSchema]. This is used when the result of the **RespondWithEnum** item can be returned in base64 format. This value is for example used when the **<ResponseItem> Id** is "KeyValue".

<StringValue> [OPTIONAL]

The **<StringValue>** element is of type **string** defined in [XMLSchema]. This is used when the **ResponseItem** element is presented in string format.

Id [Optional]

The **Id** attribute is of type **RespondWithEnum**, and it is an optional attribute. This attribute is used to identify the specified **<ResponseItem>**. See the table in 4.3.2 for a description of the different id's.

```
<!-- ResponseItem -->
<xs:element name="ResponseItem" type="ResponseItemType"/>
<xs:complexType name="ResponseItemType">
 <xs:choice>
  <xs:element name="IntValue" type="xs:integer"/>
  <xs:element name="Value" type="xs:string" maxOccurs="unbounded"/>
  <xs:element name="Base64Value" type="xs:base64Binary"
maxOccurs="unbounded"/>
  <xs:element name="StringValue" type="xs:string"/>
 </xs:choice>
 <xs:attribute name="Id" type="RespondWithEnum"/>
</xs:complexType>
```

5.4.4 Element <ContentVerifyInfo>

The **<ContentVerifyInfo>** element is used to return information about all the signatures and the belonging certificates used in the signed document. The element is REQUIRED in a response using this profile. The element is of type **ContentVerifyInfoType** and it consists of the following child elements and attribute:

<VerifyInfo> [REQUIRED]

The **<VerifyInfo>** element is described in more detail in section 5.4.5.

<Reason> [OPTIONAL]

The **<Reason>** element is only used when the **OverallAssertionStatus** attribute is set to either **NotTrusted** or **Indeterminate**. The element will point to the signature(s) that failed verification, and this MAY be done using the **Id** attribute in the **<VerifyInfo>** element like this:

<Reason> ID1, ID3 </Reason> - where the first and third signatures fails verification. ID1 and ID3 are retrieved from the **Id** attribute in the different **<VerifyInfo>** elements used in the specific verification transaction.

`OverallAssertionStatus` [REQUIRED]

The **`OverallAssertionStatus`** attribute is of type **`ContentStatusEnum`**. This attribute provides the overall result of the signature verification of the signed document. The table below lists the different status messages and their meaning.

```
<!-- ContentStatusEnum -->
<xs:simpleType name="ContentStatusEnum">
 <xs:restriction base="xs:string">
  <xs:enumeration value="Trusted"/>
  <xs:enumeration value="NotTrusted"/>
  <xs:enumeration value="Indeterminate"/>
 </xs:restriction>
</xs:simpleType>
```

ContentStatusEnum	Meaning
`Trusted`	The overall result is trusted. All signature verifications succeeded.
`NotTrusted`	The overall result is not trusted. One or more of the signatures failed verification.
`Indeterminate`	The overall result is indeterminate. The verification process could not determine if this is a trusted or not trusted request.

The following rules apply regarding the usage of **`OverallAssertionStatus`**:

- If any of the signatures have the attribute **`AssertionStatus`** set to **`Invalid`**, the **`OverallAssertionStatus`** SHALL be set to **`NotTrusted`**
- If one or more **`Indeterminate`** signatures are present, and none **`Invalid`**, the OverallAssertionStatus attribute SHALL be set to **`Indeterminate`**
- If one or more **`InsufficientQuality`** signatures are present, and none **`Invalid`** and **`Indeterminate`**, the **`OverallAssertionStatus`** attribute SHALL be set to **`NotTrusted`**.
- If all signatures have the attribute **`AssertionStatus`** set to **`Valid`**, the **`OverallAssertionStatus`** SHALL be set to **`Trusted`**

```
<!-- ContentVerifyInfo -->
<xs:element name="ContentVerifyInfo" type="ContentVerifyInfoType"/>
 <xs:complexType name="ContentVerifyInfoType">
  <xs:sequence>
   <xs:element ref="VerifyInfo" maxOccurs="unbounded"/>
   <xs:element name="Reason" type="xs:string" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="OverallAssertionStatus" type="ContentStatusEnum"
use="required"/>
 </xs:complexType>
```

5.4.5 Element <VerifyInfo>

The **`<VerifyInfo>`** element is of type **`VerifyInfoType`**, and this is a REQUIRED element in the extended validation profile of DSS. Each signature in the signed document will have its own

<VerifyInfo> element, and this element will be returned as many times as there are signatures on the signed document. The **<VerifyInfo>** consist of the following elements and attributes:

<QualityLevel> [REQUIRED]

See 5.4.2 for a detailed explanation of this element.

<ResponseItem> [REQUIRED]

See 5.4.3 for a detailed explanation of this element.

<FailureReason> [OPTIONAL]

The **<FailureReason>** element will be used if the **AssertionStatus** attribute is either **Invalid**, **Indeterminate** or **InsufficientQuality**. The element SHOULD return a code explaining the reason for failure.

Id [Required]

The **Id** attribute is used to identify the different signatures used on the signed document, hence it is the id of the signature.

AssertionStatus [Required]

The **AssertionStatus** element gives the status for every signature verification.

```
<!-- SignatureStatusEnum -->
<xs:simpleType name="SignatureStatusEnum">
 <xs:restriction base="xs:string">
  <xs:enumeration value="Valid"/>
  <xs:enumeration value="Invalid"/>
  <xs:enumeration value="Indeterminate"/>
  <xs:enumeration value="InsufficientQuality"/>
 </xs:restriction>
</xs:simpleType>
```

SignatureStatusEnum	Meaning
Valid	The signature is valid.
Invalid	The signature is invalid. The failure reason will be given in the **<FailureReason>** element
Indeterminate	It is not possible to determine the status of the signature due to for example no available CRL or OCSP responder.
InsufficientQuality	The signature is valid, but the quality of either the certificate used and/or the signature algorithms (hash and public key) are lower than the requested quality.

```
<!-- VerifyInfo -->
<xs:element name="VerifyInfo" type="VerifyInfoType"/>
<xs:complexType name="VerifyInfoType">
 <xs:sequence>
  <xs:element ref="QualityLevel"/>
  <xs:element ref="ResponseItem" maxOccurs="unbounded"/>
  <xs:element name="FailureReason" type="xs:string" minOccurs="0"/>
```

```
 </xs:sequence>
 <xs:attribute name="Id" type="xs:string" use="required"/>
 <xs:attribute name="AssertionStatus" type="SignatureStatusEnum"
use="required"/>
</xs:complexType>
```

5.4.6 Element <VerificationTimeInfo>

The element **<VerificationTimeInfo>** is already defined in [DSSCore]. This element is made required in this profile.

The **<VerificationTimeInfo>** consists of the two elements **<VerificationTime>** and **<AdditionalTimeInfo>**. The **<VerificationTime>** is REQUIRED and it is also used here as described in the [DSSCore]. The **<AdditionalTimeInfo>** element is made REQUIRED as well in this profile. This element shall be populated with the time the response message was formed while the **<VerificationTime>** is the time of verification.

New value for the **Type** attribute in **<AdditionalTimeInfo>**:

urn:oasis:names:tc:dss:1.0:additionaltimeinfo:responseTimeInstant

The **Ref** attribute are not used in this profile.

6 Profile Bindings

6.1 Transport Bindings

This profile SHOULD support both HTTP POST and SOAP 1.2 transport bindings as defined in [DSSCore].

6.2 Security Bindings

6.2.1 Security Requirements

The profile MUST authenticate the requester to the DSS Server and the DSS Server to the requester. The DSS server supporting this profile MUST support signed requests and responses to and from the service to maintain the integrity of the request and response, but only signed responses are mandatory to use.

6.2.2 TLS X.509 Mutual Authentication

This profile is secured using the TLS X.509 Mutual Authentication Binding defined in [DSSCore].

7 References

[DSSCore] S.Drees et al., Digital Signature Service Core Protocols and Elements OASIS, February 2007, http://docs.oasis-open.org/dss/v1.0/oasis-dss-core-spec-v1.0-os.html

[DSS-MultiSign] D.Hühnlein et al., Profile for Comprehensive Multi-signature Verification Reports for OASIS Digital Signature Services Version 1.0, OASIS DSS-X Committee Draft, 19 July 2009. http://docs.oasis-open.org/dss-x/profiles/verificationreport/oasis-dssx-1.0-profiles-vr-cd01.pdf

[DSS-SigPolicy] J.c.Cruellas et al., Signature Policy Profile of the OASIS Digital Signature Service Version 1.0, OASIS DSS-X Committee Draft, 18 May 2009. http://docs.oasis-open.org/dss-x/profiles/sigpolicy/oasis-dssx-1.0-profiles-sigpolicy-cd01.pdf

[XKMS] XML Key Management Specification (XKMS 2.0) Version 2.0, W3C Recommendation, 28 June 2005, http://www.w3.org/TR/2005/REC-xkms2-20050628/

[XMLDSIG] World Wide Web Consortium. XML-Signature Syntax and Processing (Second Edition), W3C Recommendation, 10 June 2008; http://www.w3.org/TR/xmldsig-core/

[XMLSchema] World Wide Web Consortium. XML Schema, Parts 0, 1, and 2 (Second Edition). W3C Recommendation, 28 October 2004; http://www.w3.org/TR/xmlschema-0/, http://www.w3.org/TR/xmlschema-1/, and http://www.w3.org/TR/xmlschema-2/

[RFC2119] S.Bradner, Key words for use in RFCs to Indicate Requirement Levels, IETF RFC2119, March 1997

8 Appendix: XML Schema

The extended validation profile of the OASIS DSS version 1.0 is attached here.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:oasis:names:tc:dss:1.0:profiles:VA:schema#"
xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
targetNamespace="urn:oasis:names:tc:dss:1.0:profiles:VA:schema#"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
schema.xsd"/>
   <xs:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/xml.xsd"/>
   <xs:import namespace="urn:oasis:names:tc:dss:1.0:core:schema"
schemaLocation="http://docs.oasis-open.org/dss/v1.0/oasis-dss-core-schema-
v1.0-os.xsd"/>
   <xs:import namespace="urn:oasis:names:tc:SAML:1.0:assertion"
schemaLocation="http://www.oasis-
open.org/committees/download.php/3408/oasis-sstc-saml-schema-protocol-
1.1.xsd"/>
   <!-- QualityLevel -->
   <xs:element name="QualityLevelRequirements" type="QualityLevelType"/>
   <xs:element name="QualityLevel" type="QualityLevelType"/>
   <xs:complexType name="QualityLevelType">
      <xs:sequence>
         <xs:element name="CertificateQuality" type="xs:string"/>
         <xs:element name="IndependentAssurance" type="xs:string"/>
         <xs:element name="HashAlgQuality" type="xs:string"/>
         <xs:element name="PublicKeyAlgQuality" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
   <!-- RespondWith -->
   <xs:element name="RespondWith" type="RespondWithEnum"/>
   <xs:simpleType name="RespondWithEnum">
      <xs:restriction base="xs:string">
         <xs:enumeration value="Subject"/>
         <xs:enumeration value="IssuerName"/>
         <xs:enumeration value="CertificateSerialNumber"/>
         <xs:enumeration value="KeyValue"/>
         <xs:enumeration value="HashAlgorithm"/>
         <xs:enumeration value="X509CertificateChain"/>
         <xs:enumeration value="X509CRL"/>
         <xs:enumeration value="SKI"/>
         <xs:enumeration value="OCSP"/>
         <xs:enumeration value="Timestamp"/>
         <xs:enumeration value="SignHash"/>
         <xs:enumeration value="ContentHash"/>
         <xs:enumeration value="Content"/>
         <xs:enumeration value="KeyUsage"/>
         <xs:enumeration value="ExtendedKeyUsage"/>
         <xs:enumeration value="BasicConstraints"/>
         <xs:enumeration value="ValidFrom"/>
         <xs:enumeration value="ValidTo"/>
         <xs:enumeration value="CRLUrl"/>
         <xs:enumeration value="CRLNumber"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:element name="ResponseItem" type="ResponseItemType"/>
   <xs:complexType name="ResponseItemType">
```

```
      <xs:choice>
         <xs:element name="IntValue" type="xs:integer"/>
         <xs:element name="Value" type="xs:string" maxOccurs="unbounded"/>
         <xs:element name="Base64Value" type="xs:base64Binary"
maxOccurs="unbounded"/>
         <xs:element name="StringValue" type="xs:string"/>
      </xs:choice>
      <xs:attribute name="Id" type="RespondWithEnum"/>
   </xs:complexType>
   <!-- Identity -->
   <xs:element name="ResponderIdentity" type="IdentityType"/>
   <xs:element name="GatewayRequesterIdentity" type="IdentityType"/>
   <xs:complexType name="IdentityType">
      <xs:sequence>
         <xs:element name="Name" type="saml:NameIdentifierType"/>
         <xs:element name="SupportingInfo" type="dss:AnyType"
minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
   <!--ContentVerifyInfo -->
   <xs:simpleType name="SignatureStatusEnum">
      <xs:restriction base="xs:string">
         <xs:enumeration value="Valid"/>
         <xs:enumeration value="Invalid"/>
         <xs:enumeration value="Indeterminate"/>
         <xs:enumeration value="InsufficientQuality"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="ContentStatusEnum">
      <xs:restriction base="xs:string">
         <xs:enumeration value="Trusted"/>
         <xs:enumeration value="NotTrusted"/>
         <xs:enumeration value="Indeterminate"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:element name="VerifyInfo" type="VerifyInfoType"/>
   <xs:complexType name="VerifyInfoType">
      <xs:sequence>
         <xs:element ref="QualityLevel"/>
         <xs:element ref="ResponseItem" maxOccurs="unbounded"/>
         <xs:element name="FailureReason" type="xs:string" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="Id" type="xs:string" use="required"/>
      <xs:attribute name="AssertionStatus" type="SignatureStatusEnum"
use="required"/>
   </xs:complexType>
   <xs:element name="ContentVerifyInfo" type="ContentVerifyInfoType"/>
   <xs:complexType name="ContentVerifyInfoType">
      <xs:sequence>
         <xs:element ref="VerifyInfo" maxOccurs="unbounded"/>
         <xs:element name="Reason" type="xs:string" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="OverallAssertionStatus" type="ContentStatusEnum"
use="required"/>
   </xs:complexType>
</xs:schema>
```