Digital Signature Service Core
Protocols and Elements

Working Draft 06, 12 November 2003

Document identifier:
oasis-dss-1.0-core-spec-wd-06

Location:
http://www.oasis-open.org/committees/dss

Editor:
Trevor Perrin, individual <trevp@trevp.net>

Contributors:
Dimitri Andivahis, Surety
Juan Carlos Cruellas, individual
Frederick Hirsch, Nokia
Pieter Kasselman, Baltimore
Andreas Kuehne, individual
John Messing, individual
Tim Moses, Entrust
Nick Pope, individual
Rich Salz, DataPower
Ed Shallow, Universal Postal Union

Abstract:
This draft defines XML request/response protocols for signing, verifying, and time-stamping of XML documents and other data. It also defines an XML time-stamp format, and an XML signature property through which a signature server can represent the client’s identity.

Status:
This is a Working Draft produced by the OASIS Digital Signature Service Technical Committee. Committee members should send comments on this draft to dss@lists.oasis-open.org.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Digital Signature Service TC web page at http://www.oasis-open.org/committees/dss/ipr.php.
# Table of Contents

1 Introduction ........................................................................................................................ 4
   1.1 Notation .......................................................................................................................... 4
   1.2 Schema Organization and Namespaces .......................................................................... 4
   1.3 DSS Overview (Non-normative) .................................................................................... 5

2 Common Protocol Structures .............................................................................................. 6
   2.1 Schema Header and Namespace Declarations ................................................................. 6
   2.2 Type <AnyType> ............................................................................................................. 6
   2.3 Type <NameType> ......................................................................................................... 6
   2.4 Element <InputDocuments> ............................................................................................ 7
      2.4.1 Commonality between <Document> and <DocumentHash> .................................. 7
      2.4.2 Element <Document> ............................................................................................ 7
      2.4.3 Element <DocumentHash> ...................................................................................... 8
   2.5 Element <Signature> ...................................................................................................... 9
   2.6 Elements <Options> and <Outputs> ............................................................................. 10
   2.7 Element <Result> .......................................................................................................... 10

3 The DSS Signing Protocol ................................................................................................. 12
   3.1 Element <SignRequest> .................................................................................................. 12
   3.2 Element <SignResponse> ............................................................................................... 12
   3.3 Basic Processing ........................................................................................................... 13
      3.3.1 Enveloping Signatures .......................................................................................... 13
      3.3.2 Enveloped Signatures ........................................................................................... 13
   3.4 Options and Outputs ..................................................................................................... 13
      3.4.1 Option <ServiceProfile> ....................................................................................... 13
      3.4.2 Option <ServicePolicy> ....................................................................................... 14
      3.4.3 Option <ClaimedIdentity> .................................................................................... 14
      3.4.4 Options <SignatureTimestamp> and <ContentTimestamp> ................................... 14
      3.4.5 Option <IntendedAudience> ................................................................................. 14
      3.4.6 Option <KeySelector> ......................................................................................... 14
      3.4.7 Option <SignedReferences> .................................................................................. 15
      3.4.8 Option <Properties> ............................................................................................ 15
      3.4.9 Option <SignaturePlacement> and Output <OutputDocument> ............................... 16

4 The DSS Verifying Protocol ............................................................................................... 18
   4.1 Element <VerifyRequest> ............................................................................................... 18
   4.2 Element <VerifyResponse> ............................................................................................ 18
   4.3 Basic Processing .......................................................................................................... 19
   4.4 Result Codes ................................................................................................................ 19
   4.5 Options and Outputs .................................................................................................... 20
      4.5.1 Option <ServiceProfile> ...................................................................................... 20
      4.5.2 Option <ServicePolicy> ...................................................................................... 20
1 Introduction

This specification defines the XML syntax and semantics for the Digital Signature Service core protocols, and for some associated core elements. The core protocols support signing, verifying, and time-stamping of XML documents and other data. The core elements extend XML Signatures [XMLSig] to contain time-stamps and representations of the client's identity.

The core protocol messages are typically bound into other structures for transport, such as XML-encoded SOAP messages. The core protocols are also typically profiled to constrain optional features and add additional features. A companion document provides an initial set of bindings and profiles [DSSBind]. A file containing just the core schema [Core-XSD] is also available.

The following sections describe how to understand the rest of this specification.

1.1 Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF RFC 2119 [RFC2119]. These keywords are capitalized when used to unambiguously specify requirements over protocol and application features and behaviour that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

Listings of DSS schemas appear like this.

Example code listings appear like this.

In cases of disagreement between the the DSS schema file [Core-XSD] and this specification, the schema file takes precedence.

Conventional XML namespace prefixes are used throughout the listings in this specification to stand for their respective namespaces (see Section 1.2) as follows, whether or not a namespace declaration is present in the example:

- The prefix dss: stands for the DSS namespace.
- The prefix ds: stands for the W3C XML Signature namespace [XMLSig-XSD].
- The prefix xs: stands for the W3C XML Schema namespace [Schema1].

This specification uses the following typographical conventions in text: <DSSElement>, <ns:ForeignElement>, Attribute, Datatype, OtherCode.

1.2 Schema Organization and Namespaces

The DSS core structures are defined in a schema [Core-XSD] associated with the following XML namespace:

http://www.oasis-open.org/tc/DSS/1.0/core/schema

Imported into this schema is the schema for XML Signature [XMLSig-XSD], which is associated with the following XML namespace:

http://www.w3.org/2000/09/ xmldsig#
1.3 DSS Overview (Non-normative)

This specification outlines an XML-based protocol and common XML schema structures necessary to support a delegated XML signing and verification service, as well as a time stamping service. One of the goals is to define an XML-based protocol that can support a variety of signature and timestamp server implementations, supporting both XML signature and non-XML signature services, for example, with a single XML-based protocol. Application profiles and server implementations may constrain what is supported in a specific deployment. The protocol and core elements are designed to be flexible and extensible through the use of an open XML schema.

There are two major services supported by this specification – signature and time stamp services. Signature services include signing and verification and may include time mark attributes in signatures. Time mark signature attributes may include the time of signing, for example. The time stamp service is different in that it defines requests, responses, and XML schema formats to support authoritative timestamps analogous to the TimeStampToken defined in RFC 3161, supporting proof that a datum existed before a specific point in time.

Options are used to allow a variety of complex choices to be uniformly expressed and managed. Signing options include the kind of signature to be returned (e.g., CMS, XML Signature), how an XML signature is to be delivered (detached, enveloped, enveloping), where it is to be placed in a document when not detached, and which signature attributes are to be generated by the server, for example. It is expected that this specification will be profiled to constrain the options and define necessary extensions to support specific applications in an interoperable manner. This specification assumes that protocol requests either succeed or fail, avoiding the complexity of partial success when some options are not met. It is anticipated that application profiles will define meaningful sets of supported options and appropriate defaults.

One example of a possible profile is a DSS Web Service Security Profile, defining how the DSS protocol may be used to request SOAP Message security headers containing XML Signatures and the supporting tokens used to convey the corresponding keys. Another example would be a “Corporate Seal” profile, defining how the protocol and structures may be used to support an application that uses a single corporate key to sign documents for data origin authentication.

One of the specific goals of this specification is to support interoperation between DSS server implementations as well as interoperability of the signatures and timestamps between DSS-based and non-DSS aware signature and timestamp implementations.

How this protocol is used with an underlying protocol is defined by the appropriate protocol bindings document. Use with SOAP, for example, will be defined in the DSS SOAP Bindings specification.

This specification does not define general policy mechanisms, but does define interoperable means to specify policies in requests and signatures, using policy QNames that can also be used to specify OIDs. Explicit processing steps may be specified in requests such as “verify an existing signature” before countersigning.

Return options include the ability to return supporting information (such as OCSP responses) as well as the information on processing steps followed enabling a client to verify correct operation. A DSS server ultimately determines what actions are taken according to an application profile.
2 Common Protocol Structures

2.1 Schema Header and Namespace Declarations

The following schema fragment defines the XML namespaces and other header information for
the DSS schema:

```xml
<xs:schema
    targetNamespace="http://www.oasis-open.org/tc/DSS/1.0/core/schema"
    xmlns:dss="http://www.oasis-open.org/tc/DSS/1.0/core/schema"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">
```

2.2 Type <AnyType>

The AnyType complex type allows arbitrary XML content within an element:

```xml
<xs:complexType name="AnyType">
    <xs:sequence>
        <xs:any processContents="lax" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

2.3 Type <NameType>

The NameType complex type is used where different types of names are needed:

```xml
<xs:complexType name="NameType">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="Format" type="xs:QName"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

The Format attribute contains an XML Schema QName which determines how the string content
is interpreted. A namespace prefix for the QName MUST be provided. The QName may be a
value defined by this specification, or a value defined by some other specification, in some other
namespace. The values defined by this specification are:

- **EmailAddress**
  Indicates that the string content is in the form of an email address, specifically “addr-spec” as
defined in [RFC 2822]. An addr-spec has the form local-part@domain.

- **X509SubjectName**
  Indicates that the string content is in the form specified for the contents of the
  `<ds:X509SubjectName>` element in [XMLSig].

- **URI**
  Indicates that the string content is in the form of a URI as defined in [RFC 2396]. The URI
  MUST be absolute, and MAY have a fragment identifier (i.e., it may be a URI Reference).
2.4 Element <InputDocuments>

The <InputDocuments> element is used to send documents to a DSS server, whether for signing, verifying, or time-stamping. It consists of any number of the following elements:

- <Document> [Any Number]
  - An XML document or some other data.
- <DocumentHash> [Any Number]
  - A hash value of an XML document or some other data.

The following schema fragment defines the <InputDocuments> element:

```xml
<xs:element name="InputDocuments">
  <xs:complexType>
    <xs:sequence>
      <xs:choice maxOccurs="unbounded">
        <xs:element ref="dss:Document"/>
        <xs:element ref="dss:DocumentHash"/>
        <xs:any processContents="lax"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

2.4.1 Commonality between <Document> and <DocumentHash>

Both the <Document> and <DocumentHash> elements contain the following attributes and child elements:

- ID [Optional]
  - The identifier used to refer to this input document within the protocol message.
- RefURI [Optional]
  - This specifies the value for the <ds:Reference> element’s URI attribute when referring to this input document.
- RefType [Optional]
  - This specifies the value for the <ds:Reference> element’s Type attribute when referring to this input document.
- <ds:Transforms> [Optional]
  - This specifies the value for the <ds:Reference> element’s <ds:Transforms> child element when referring to this input document.

2.4.2 Element <Document>

The <Document> element may contain the following elements (in addition to the common ones listed in 2.2.1):

- <XMLData> [Optional]
  - This contains arbitrary XML content.
- <Base64Data> [Optional]
  - This contains a base64 encoding of an XML document or some other data. The type of data is specified by its MimeTyp attribute.
The following schema fragment defines the `<Document>, <XMLData>, and <Base64Data>` elements:

```
<xs:element name="Document">
  <xs:complexType>
    <xs:sequence>
      <xs:choice>
        <xs:element ref="dss:XMLData"/>
        <xs:element ref="dss:Base64Data"/>
      </xs:choice>
      <xs:element ref="ds:Transforms" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="ID" type="xs:ID" use="optional"/>
    <xs:attribute name="RefURI" type="xs:anyURI" use="optional"/>
    <xs:attribute name="RefType" type="xs:anyURI" use="optional"/>
  </xs:complexType>
</xs:element>
```

2.4.3 Element `<DocumentHash>`

The `<DocumentHash>` element contains the following elements (in addition to the common ones listed in 2.2.1):

`<ds:DigestMethod>` [Required]
This identifies the digest algorithm used to hash the document.

`<ds:DigestValue>` [Required]
This gives the document’s hash value.

The following schema fragment defines the `<DocumentHash>` element:

```
<xs:element name="Document">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ds:DigestMethod"/>
      <xs:element ref="ds:DigestValue"/>
      <xs:element ref="ds:Transforms" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="ID" type="xs:ID" use="optional"/>
    <xs:attribute name="RefURI" type="xs:anyURI" use="optional"/>
    <xs:attribute name="RefType" type="xs:anyURI" use="optional"/>
  </xs:complexType>
</xs:element>
```
2.5 Element <Signature>

The <Signature> element is returned in a sign response, and sent in a verify request. It may contain one of the following elements:

- <ds:Signature> [Optional]
  - An XML Signature [XMLSig].
- <Base64Signature> [Optional]
  - A base64 encoding of some non-XML signature, such as a PGP [PGP] or CMS [CMS] signature. The type of signature is specified by its MimeType attribute.
- <SignaturePtr> [Optional]
  - This points to an XML Signature that may be in one of the input documents, or one of the outputs.
  - A <SignaturePtr> contains the following elements:
    - WhichDocument [Required]
      - This identifies the document being pointed at.
    - XPath [Optional]
      - This identifies the element being pointed at. It may be omitted if there is only a single <ds:Signature> element in the pointed-to document.

The following schema fragment defines the <Signature>, <Base64Signature>, and <SignaturePtr> elements:

```
<xs:element name="Signature">
  <xs:complexType>
    <xs:choice>
      <xs:element ref="ds:Signature"/>
      <xs:element ref="dss:Base64Signature"/>
      <xs:element ref="dss:SignaturePtr"/>
      <xs:any processContents="lax"/>
    </xs:choice>
  </xs:complexType>
</xs:element>

<xs:element name="Base64Signature">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:base64Binary">
        <xs:attribute name="MimeType" type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="SignaturePtr">
  <xs:complexType>
    <xs:attribute name="WhichDocument" type="xs:IDREF"/>
    <xs:attribute name="XPath" type="xs:string"/>
  </xs:complexType>
</xs:element>
```
2.6 Elements <Options> and <Outputs>

All request messages can contain an <Options> element, and all response messages can contain an <Outputs> element. The <Options> contains all options about the request. Profiles will specify which options are allowed and what their default values are. All options SHOULD have some default value, so that a client may omit the <Options> element yet still get service from any DSS server. If a server doesn’t recognize or can’t handle any option, it MUST reject the request outright. Options can appear in any order within the <Options> element.

The <Outputs> element contains additional protocol outputs. The client MAY request the server to respond with certain outputs by sending certain options. The server MAY also respond with outputs the client didn’t request, depending on the server’s profile and policy. Outputs can appear in any order within the <Outputs> element.

The following schema fragment defines the <Options> and <Outputs> elements:

```
<xs:element name="Options" type="dss:AnyType"/>
<xs:element name="Outputs" type="dss:AnyType"/>
```

2.7 Element <Result>

The <Result> element is returned with every response message. It contains the following elements:

- `<ResultMajor>` [Required]
  - The most significant component of the result code.
- `<ResultMinor>` [Optional]
  - The least significant component of the result code.
- `<ResultMessage>` [Optional]
  - A message which MAY be returned to an operator.

The following schema fragment defines the <Result> element:

```
<xs:element name="Result">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ResultMajor" type="xs:QName"/>
      <xs:element name="ResultMinor" type="xs:QName" minOccurs="0"/>
      <xs:element name="ResultMessage" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The <ResultMajor> and <ResultMinor> elements each contain an XML Schema QName. A namespace prefix MUST be provided. The <ResultMajor> QName MUST be one of these values:

- **Success**
  - The protocol executed successfully.
- **Requester**
  - The request could not be performed due to an error on the part of the requester.
The request could not be performed due to an error on the part of the responder.
The `<ResultMinor>` QName may be a value defined by this specification, or a value defined by
some other specification, in some other namespace.
The Requester `<ResultMajor>` code may be followed by:
Not Authorized
The client is not authorized to perform the request.
Not Supported
The server didn’t recognize or doesn’t support some aspect of the request.
The Success `<ResultMajor>` code on a verify response message SHALL be followed by a
`<ResultMinor>` code which indicates the status of the signature. See section 4 for details.
3 The DSS Signing Protocol

3.1 Element <SignRequest>

The <SignRequest> element is sent by the client to request a signature on some input documents. It contains the following elements and attributes:

- <Options> [Optional]
  This element contains all options about the request.

- <InputDocuments> [Required]
  A list of input documents which the signature will be calculated over.

- RequestID [Optional]
  This attribute is used to correlate requests with responses. When present in a request, the server MUST return it in the response.

The following schema fragment defines the <SignRequest> element:

```xml
<xs:element name="SignRequest">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="dss:Options" minOccurs="0"/>
      <xs:element ref="dss:InputDocuments"/>
    </xs:sequence>
    <xs:attribute name="RequestID" type="xs:string" use="optional"/>
  </xs:complexType>
</xs:element>
```

3.2 Element <SignResponse>

The <SignResponse> element contains:

- <Result> [Required]
  A code representing the status of the request.

- <Signature> [Optional]
  The resultant signature, if the request succeeds.

- <Outputs> [Optional]
  Any additional outputs returned by the server.

- RequestID [Optional]
  This attribute is used to correlate requests with responses. When present in a request, the server MUST return it in the response.

The following schema fragment defines the <SignResponse> element:
3.3 Basic Processing

With no options, a server receiving a `<SignRequest>` proceeds as follows:

1. The server hashes each input `<Document>`.
2. The server forms a `<ds:Reference>` for each input document out of its `RefURI`, `RefType`, `<ds:Transforms>`, and hash value.
3. The server forms a `<ds:SignedInfo>` out of the `<ds:Reference>` elements.
4. The server forms a `<ds:Signature>` by signing the `<ds:SignedInfo>`.
5. The server returns the `<ds:Signature>`.

Additional processing may be carried out as specified by the options, or as implied by the profile the server is operating under.

3.3.1 Enveloping Signatures

To create an XML Signature that envelopes one or more of the input documents, the client simply splices the appropriate input document(s) into the returned `<ds:Signature>`.

3.3.2 Enveloped Signatures

To create an XML Signature that is enveloped by one of the input documents, the client simply indicates an Enveloped Signature Transform [XMLSig] on the appropriate input document, and splices the returned `<ds:Signature>` into the Input Document.

3.4 Options and Outputs

This document defines some options and outputs that might be useful in multiple profiles. Profiles can define their own options and outputs, as well. Handling of these is discussed in 2.5.

3.4.1 Option `<ServiceProfile>`

The `<ServiceProfile>` element indicates a particular profile. This may be used to select a profile if a server supports multiple profiles, or as a sanity-check to make sure the server implements the profile the client thinks he does.

The following schema fragment defines the `<ServiceProfile>` element:

```xml
<xs:element name="ServiceProfile" type="xs:anyURI"/>
```
3.4.2 Option <ServicePolicy>

The <ServicePolicy> element indicates a particular policy associated with the DSS service. The policy may include information on the characteristics of the server that are not necessarily covered by the <ServiceProfile> element. This may be used to select a specific policy if a service supports multiple policies for a specific profile, or as a sanity-check to make sure the server implements the policy the client thinks he does.

The following schema fragment defines the <ServicePolicy> element:

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

3.4.3 Option <ClaimedIdentity>

The <ClaimedIdentity> element indicates the identity of the client who is requesting the signature. The server should check this against the client’s authentication credentials, and then may use this to parameterize any aspect of its processing.

The following schema fragment defines the <ClaimedIdentity> element:

```xml
<xs:element name="ClaimedIdentity" type="dss:NameType"/>
```

3.4.4 Options <SignatureTimestamp> and <ContentTimestamp>

The <SignatureTimestamp> and <ContentTimestamp> elements are boolean flags that indicate the client wishes the server to provide the appropriate type of timestamp as a signature attribute. Both flags may be present simultaneously.

The following schema fragment defines both elements:

```xml
<xs:element name="SignatureTimestamp"/>
<xs:element name="ContentTimestamp"/>
```

3.4.5 Option <IntendedAudience>

The <IntendedAudience> element tells the server who the signature is meant for.

The following schema fragment defines the <IntendedAudience> element:

```xml
<xs:element name="IntendedAudience">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Recipient" type="dss:NameType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

3.4.6 Option <KeySelector>

The <KeySelector> element tells the server which key to use.

The following schema fragment defines the <KeySelector> element:

```xml
<xs:element name="KeySelector">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ds:KeyInfo"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```
3.4.7 Option <SignedReferences>

The <SignedReferences> element gives the client greater control over how the <SignedReference> elements are formed. When this element is present, the second step of "Basic Processing" is overridden, and instead each <SignedReference> element within <SignedReferences> controls the creation of a corresponding <ds:Reference>. Each <SignedReference> element contains:

- **WhichDocument** [Required]
  - Which input document this reference refers to.
- **<RefId>** [Optional]
  - Sets the Id attribute on the corresponding <ds:Reference>.
- **<ds:Transforms>** [Optional]
  - Requests the server to perform additional transforms on this reference.

The following schema fragment defines the <SignedReferences> element:

```xml
<xs:element name="SignedReferences">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="dss:SignedReference" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

3.4.8 Option <Properties>

The <Properties> element is used to request that the server add certain signed or unsigned properties (aka “attributes”) into the signature. The client can send the server a particular value to use for each property, or leave that up to the server to determine. The <Properties> element contains:

- **<SignedProperties>** [Optional]
  - These properties will be covered by the signature.
- **<UnsignedProperties>** [Optional]
  - These properties will not be covered by the signature.
Each `<Property>` element contains:

- **<Identifier>** [Required]
  - A QName identifying the property.
- **<Value>** [Optional]
  - If present, the value the server should use for the property.

The following schema fragment defines the `<Properties>` element:

```xml
<xs:complexType name="PropertiesType">
  <xs:sequence>
    <xs:element ref="dss:Property" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

3.4.9 Option `<SignaturePlacement>` and Output `<OutputDocument>`

The `<SignaturePlacement>` element instructs the server to place the signature inside one of the input documents, and return the resulting document. The `<SignaturePlacement>` element contains the following attributes and elements:

- **WhichDocument** [Required]
  - Identifies the input document which the signature will be inserted into.
- **<XPathAfter>** [Optional]
  - Identifies an element, in the input document, which the signature will be inserted after.
- **<XPathFirstChildOf>** [Optional]
  - Identifies an element, in the input document, which the signature will be inserted as the first child of.

The `<OutputDocument>` element contains the XML input document with the signature inserted. It has one child element:

- **<XMLData>** [Optional]
  - This contains arbitrary XML content.
The following schema fragment defines the `<SignaturePlacement>` and `<OutputDocument>` elements:

```
<xs:element name="SignaturePlacement">
  <xs:complexType>
    <xs:choice>
      <xs:element name="XPathAfter" type="xs:string"/>
      <xs:element name="XPathFirstChildOf" type="xs:string"/>
    </xs:choice>
    <xs:attribute name="WhichDocument" type="xs:IDREF"/>
  </xs:complexType>
</xs:element>

<xs:element name="OutputDocument">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="XMLData"/>
      <xs:sequence>
        <xs:element ref="XMLData"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
```
4 The DSS Verifying Protocol

4.1 Element <VerifyRequest>

The <VerifyRequest> element is sent by the client to verify a signature on some input
documents. It contains the following elements and attributes:

- **<Options>** [Optional]
  - This element contains all options about the request.

- **<Signature>** [Required]
  - This element contains a signature or points to an XML Signature in one of the Input
    Documents.

- **<InputDocuments>** [Required]
  - A lists of input documents which the signature was calculated over.

- **RequestID** [Optional]
  - This attribute is used to correlate requests with responses. When present in a request, the
    server MUST return it in the response.

The following schema fragment defines the <VerifyRequest> element:

```xml
<x:element name="VerifyRequest">
  <x:complexType>
    <x:sequence>
      <x:element ref="dss:Options" minOccurs="0"/>
      <x:element ref="dss:Signature" minOccurs="0"/>
      <x:element ref="dss:InputDocuments"/>
    </x:sequence>
    <x:attribute name="RequestID" type="xs:string" use="optional"/>
  </x:complexType>
</x:element>
```

4.2 Element <VerifyResponse>

The <VerifyResponse> element contains:

- **<Result>** [Required]
  - A code representing the status of the corresponding request.

- **<Outputs>** [Optional]
  - Any outputs that were requested by the presence of a corresponding option in the
    <VerifyRequest> message.

- **RequestID** [Optional]
  - This attribute is used to correlate requests with responses. When present in a request, the
    server MUST return it in the response.

The following schema fragment defines the <VerifyResponse> element:
4.3 Basic Processing

With no options, a server receiving a <VerifyRequest> proceeds as follows:

1. The server dereferences the <Signature> element to retrieve the [XMLSig] signature.

2. For each <ds:Reference> in the signature, the server finds the input document with matching RefURI and RefType values.

3. If the input document is a <DocumentHash>, the server checks that the <ds:Transforms>, <ds:DigestMethod>, and <ds:DigestValue> elements match between the <DocumentHash> and the <ds:Reference>.

4. If the input document is a <Document>, the server applies any transforms specified by the <ds:Reference>, and then hashes the resultant data object according to <ds:DigestMethod>, and checks that the result matches <ds:DigestValue>.

5. The server then validates the signature according to section 3.2.2 in [XMLSig].

Additional processing may be carried out as specified by the options, or as implied by the profile the server is operating under.

4.4 Result Codes

Whether the signature succeeds or fails to verify, the server will return the Success <ResultMajor> code. The <ResultMinor> QName must be one of the following values, or some other value defined by some profile of this specification:

ValidSignature

The signature is valid.

IndeterminateKey

The server could not determine whether the signing key is valid. For example, the server might not have been able to construct a certificate path to the signing key.

UntrustedKey

The signature is performed by a key the server considers suspect. For example, the signing key may have been revoked, or it may be a different key from what the server is expecting the signer to use.

IncorrectSignature

The signature fails to verify, indicating that the message was modified in transit, or that the signature was performed incorrectly.
4.5 Options and Outputs

This document defines some options and outputs that might be useful in multiple profiles. Profiles can define their own options and outputs, as well.

4.5.1 Option <ServiceProfile>

The <ServiceProfile> element indicates a particular profile. This may be used to select a profile if a server supports multiple profiles, or as a sanity-check to make sure the server implements the profile the client thinks he does.

The following schema fragment defines the <ServiceProfile> element:

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

4.5.2 Option <ServicePolicy>

The <ServicePolicy> element indicates a particular policy associated with the DSS service. The policy may include information on the characteristics of the server that are not necessarily covered by the <ServiceProfile> element. This may be used to select a specific policy if a service supports multiple policies for a specific profile, or as a sanity-check to make sure the server implements the policy the client thinks he does.

The following schema fragment defines the <ServicePolicy> element:

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

4.5.3 Option <ClaimedIdentity>

The <ClaimedIdentity> element indicates the identity of the client who is requesting the verification. The server should check this against the client’s authentication credentials, and then may use this to parameterize any aspect of its processing.

The following schema fragment defines the <ClaimedIdentity> element:

```xml
<xs:element name="ClaimedIdentity" type="dss:NameType"/>
```

4.5.4 Option <IgnoreMissingInputDocuments>

The presence of this element instructs the server not to give an error if he can’t find an input document that matches a particular <ds:Reference>, but instead to assume that the client has already validated this <ds:Reference> himself.

The following schema fragment defines the <IgnoreMissingInputDocuments> element:

```xml
<xs:element name="IgnoreMissingInputDocuments"/>
```

4.5.5 Option <VerifyManifests>

The presence of this element instructs the server to attempt to validate any input documents it encounters whose Type attribute equals http://www.w3.org/2000/09/xmldsig#Manifest. Such an input document MUST contain an XML element of type ds:ManifestType. On encountering such a document in step 2 of basic processing, the server should repeat step 2 for all the <ds:Reference> elements within the manifest.
The following schema fragment defines the <VerifyManifests> element:

```xml
<xs:element name="VerifyManifests"/>
```

### 4.5.6 Option <VerificationTime>

This element instructs the server to attempt to determine the signature’s validity at the specified time, instead of the current time.

The following schema fragment defines the <VerificationTime> element:

```xml
<xs:element name="VerificationTime" type="xs:dateTime"/>
```

### 4.5.7 Option <AdditionalKeyInfo>

This element provides the server with additional data (such as certificates and CRLs) which it can use to validate the signing key.

The following schema fragment defines the <AdditionalKeyInfo> element:

```xml
<xs:element name="AdditionalKeyInfo">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ds:KeyInfo"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

### 4.5.8 Option <ReturnProcessingDetails> and Output <ProcessingDetails>

The presence of the <ReturnProcessingDetails> option instructs the server to return a <ProcessingDetails> output, elaborating on what signature verification steps succeeded or failed. The <ProcessingDetails> element may contain the following child elements:

- **<ValidDetail>** [Any Number]
  - A verification aspect that was evaluated and found to be valid.

- **<IndeterminateDetail>** [Any Number]
  - A verification aspect that could not be evaluated or was evaluated and returned an indeterminate result.

- **<InvalidDetail>** [Any Number]
  - A verification aspect that was evaluated and found to be invalid.

These elements each contain an XML Schema QName which identifies the detail. A namespace prefix for the QName MUST be provided. The QName may be a value defined by this specification, or a value defined by some other specification, in some other namespace. The values defined by this specification are the following, interpreted as per [XKMS] section 5.18:

- IssuerTrust, RevocationStatus, ValidityInterval, Signature.

The following schema fragment defines the <ReturnProcessingDetails> and <ProcessingDetails> elements:
<xs:element name="ReturnProcessingDetails">
<xs:element name="ProcessingDetails">
<xs:complexType>
<xs:sequence>
<xs:element name="ValidDetail" type="xs:QName" minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="IndeterminateDetail" type="xs:QName" minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="InvalidDetail" type="xs:QName" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:element>

4.5.9 Option <ReturnSigningTime> and Output <SigningTime>

The presence of the <ReturnSigningTime> option instructs the server to return a <SigningTime> output. This output contains an indication of when the signature was performed, and a boolean attribute that indicates whether this value should be relied upon or not.
The following schema fragment defines the <ReturnSigningTime> and <SigningTime> elements:

<xs:element name="ReturnSigningTime"/>
<xs:element name="SigningTime">
<xs:complexType>
<xs:simpleContent>
<xs:extension base="xs:dateTime">
<xs:attribute name="Trusted" type="xs:boolean"/>
</xs:extension>
</xs:complexType>
</xs:element>

4.5.10 Option <ReturnSignerIdentity> and Output <SignerIdentity>

The presence of the <ReturnSignerIdentity> option instructs the server to return a <SignerIdentity> output. This output contains an indication of who performed the signature.
The following schema fragment defines the <ReturnSignerIdentity> and <SignerIdentity> elements:

<xs:element name="ReturnSignerIdentity"/>
<xs:element name="SignerIdentity" type="dss:Name"/>

4.5.11 Option <ReturnUpdatedSignature> and Output <UpdatedSignature>

The presence of the <ReturnUpdatedSignature> option instructs the server to return an <UpdatedSignature> output. This output contains the original signature with some additional attributes added to it.
The following schema fragment defines the <ReturnUpdatedSignature> and <UpdatedSignature> elements:
<xs:element name="ReturnUpdatedSignature"/>
<xs:element name="UpdatedSignature">
  <xs:complexType>
    <xs:element ref="dss:Signature"/>
  </xs:complexType>
</xs:element>
5 Timestamp token

This section contains the definition of the timestamp token.

5.1 Schema Header and Namespace Declarations

The following schema fragment defines the XML namespaces and other header information for the DSS schema:

```xml
<xs:schema
  targetNamespace="http://www.oasis-open.org/tc/DSS/1.0/timestamp-token/schema"
  xmlns:tst="http://www.oasis-open.org/tc/DSS/1.0/timestamp-token/schema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
```

This schema imports definitions from the XML Digital Signature schema.

5.2 Element <Tst>

The <Tst> element represents a single timestamp token.

```xml
<xs:element name="Tst" type="ds:SignatureType">
```

The <Tst> element has the same type-definition as the ds:SignatureType definition. In this way, a timestamp token can be created and validated by a conventional XML Digital Signature implementation.

The following sections define how the elements of the <ds:Signature> element MUST be used.

- **ds:KeyInfo/ [Required]**
  - The <KeyInfo> element SHALL identify the issuer of the timestamp and MAY be used to locate, retrieve and validate the timestamp token signature-verification key.

- **ds:SignedInfo/Reference [Required]**
  - The <Reference> element SHALL contain a bare name XPointer reference to the <tstInfo> element. It MUST also reference the document or documents that are timestamped.

- **ds:Object/ [Required]**
  - The <TstInfo> element SHALL be contained in an <Object> element. Any extension elements that are not defined by this specification SHALL also be represented as <Object> elements.

5.3 Element <TstInfo>

A <TstInfo> element MUST be included in the <Tst> element as a <ds:Signature/Object> element. The <TstInfo> element is of type tstInfoType.
5.4 ComplexType TstInfoType

This section contains the definition of the TstInfoType complex type.

```xml
<xs:complexType name="TstInfoType">
  <xs:attribute name="SerialNumber" type="xs:integer"/>
  <xs:attribute name="CreationTime" type="xs:dateTime"/>
  <xs:attribute name="Policy" type="xs:anyURI" use="optional"/>
  <xs:attribute name="ErrorBound" type="xs:duration"/>
  <xs:attribute name="Ordered" type="xs:boolean" default="false"/>
</xs:complexType>
```

Defines the following attributes.

SerialNumber [Required]
This attribute SHALL contain a serial number produced by the timestamp authority. It MUST be unique across all the tokens issued by a particular TSA. Provided relying parties do not accept timestamp tokens from distinct TSAs that use the same name, the combination of the issuer name and the serial number will uniquely identify a timestamp token to a particular relying party.

CreationTime [Required]
The time at which the token was issued. It SHALL be a time according to the local clock of the authority, no earlier than the time at which the request was completely received and no later than the time at which the signature process was started.

Policy [Optional]
This attribute SHALL identify the policy under which the token was issued. If the corresponding element appears in the request, then this element MUST contain one of the values supplied in the request. Amongst other things, the TSA’s policy SHOULD identify the fundamental source of its time.

ErrorBound [Optional]
The TSA’s estimate of the maximum error in its local clock.

Ordered [Default="false"]
This attribute SHALL indicate whether or not timestamps issued by this TSA, under this policy, are strictly ordered according to the value and precision of the creationTime attribute value.

5.5 Timestamp verification procedure

If any one of these steps results in failure, then the timestamp token SHOULD be rejected.

1. Locate and verify the signature-verification key corresponding to the ds:KeyInfo/element contents.
2. Verify that the signature-verification key is authorized for verifying timestamps.
3. Verify that the signature-verification key conforms with all relevant aspects of the relying-party’s policy.
4. Verify that all digest and signature algorithms conform with the relying-party’s policy.
5. Verify that the signature-verification key is consistent with the ds:SignedInfo/SignedInfo/Reference/@URI attribute whose value is "#tstInfo".
7. Verify that there is a `ds:SignedInfo/Reference/@` attribute that correctly identifies the timestamped document.

8. Verify that the `tstInfo/@policy` attribute value is acceptable.

9. Establish a maximum acceptable error bound value and verify that the `tstInfo/@errorBound` attribute value is less than or equal to this value.

10. Verify all digests and the signature.

11. If comparing the `tstInfo/@creationTime` attribute value to another time value, first verify that they differ by more than the maximum acceptable error bound value.
6 Editorial Issues

1) Another way of handling the options is to have each option placed within an `<Option>` element. This has the advantage that each option could be tagged with a `mustUnderstand` attribute, so the server would know whether it was okay to ignore the option or not. It has the disadvantage of making things a little more verbose.

*Resolution:* Leave as is, per 10/20/2003 meeting.

2) It is suggested that the RequestID option be put in the top level of the protocol structure so that it can be used at the basic level of the DSS protocol handler.

*Resolution:* This has been done, per 10/20/2003 meeting.

3) The utility of the `<DocumentURI>` element has been questioned.

*Resolution:* Since Rich, John, Trevor, and perhaps Andreas seem in favor of removing this, and only Gregor and Juan Carlos, and perhaps Nick, seem in favor of keeping it, it’s been removed.

4) Should every Output only be returned if the client requests it, through an Option?

*Resolution:* No - Servers can return outputs on their own initiative, per 11/3/2003 meeting.

5) Should Signature Placement, and elements to envelope, be made Signature Options?

*Resolution:* Yes - per 11/3/2003 meeting, but hasn’t been done yet.

6) Should `<Options>` be renamed? To `<AdditionalInputs>`, `<Inputs>`, `<Parameters>`, or something else?

7) Should we adopt a Timestamp more like Dimitri’s `<Tst>`?

8) The `<ProcessingDetails>` are a little sketchy, these could be fleshed out.

9) A `<dss:Signature>` can contain a `<dss:SignaturePtr>`, which uses an XPath expression to point to a signature. This allows a client to send an `<InputDocument>` to the server with an embedded signature, and just point to the signature, without copying it. Is it acceptable to require all servers to support XPath, for this?
7 References

7.1 Normative


## Appendix A. Revision History

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>By Whom</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td>wd-01</td>
<td>2003-10-03</td>
<td>Trevor Perrin</td>
<td>Initial version</td>
</tr>
<tr>
<td>wd-02</td>
<td>2003-10-13</td>
<td>Trevor Perrin</td>
<td>Skeleton of verify as well</td>
</tr>
<tr>
<td>wd-03</td>
<td>2003-10-19</td>
<td>Trevor Perrin</td>
<td>Added TimeStampToken, References</td>
</tr>
<tr>
<td>wd-04</td>
<td>2003-10-29</td>
<td>Trevor Perrin</td>
<td>Fleshed things out</td>
</tr>
<tr>
<td>wd-05</td>
<td>2003-11-9</td>
<td>Trevor Perrin</td>
<td>Added Name, clarified options-handling</td>
</tr>
<tr>
<td>wd-06</td>
<td>2003-11-12</td>
<td>Trevor Perrin</td>
<td>Added more options/outputs</td>
</tr>
</tbody>
</table>
Appendix B. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.

Copyright © OASIS Open 2003. All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself does not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.