Digital Signature Service Core
Protocols and Elements

Working Draft 02, 12 October 2003

Document identifier: oasis-dss-1.0-core-spec-wd-02
Location: http://www.oasis-open.org/committees/dss
Editor: Trevor Perrin, individual <trevp@trevp.net>
Contributors:
Juan Carlos Cruellas, individual
Frederick Hirsch, Nokia
Andreas Kuehne, 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.
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 protocols 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 Element <InputDocuments>

The <InputDocuments> element is used in request messages 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.
- <DocumentURI> [Any Number]
  A URI pointing to 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:DocumentURI"/>
        <xs:element ref="dss:DocumentHash"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

2.2.1 Commonality amongst <Document*> Elements

Every <Document*> element contains the following attributes and elements:

- ID [Required]
  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 document.

<ds:Transforms> [Optional]

This specifies the value for the <ds:Reference> element's <ds:Transforms> child element when referring to this document.

2.2.2 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 MimeType attribute.

The following schema fragment defines the Document, XMLData, and Base64Data elements:

```xml
<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"/>
    <xs:attribute name="RefURI" type="xs:anyURI" use="optional"/>
    <xs:attribute name="RefType" type="xs:anyURI" use="optional"/>
  </xs:complexType>
</xs:element>

<xs:element name="XMLData">
  <xs:complexType>
    <xs:sequence>
      <xs:any processContents="lax" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Base64Data">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:base64Binary">
        <xs:attribute name="MimeType" type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```
2.2.3 Element <DocumentURI>

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

<URI> [Required]

A URI which the server will dereference to retrieve the input document.

The following schema fragment defines the <DocumentURI> element:

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

2.2.4 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"/>
    <xs:attribute name="RefURI" type="xs:anyURI" use="optional"/>
    <xs:attribute name="RefType" type="xs:anyURI" use="optional"/>
  </xs:complexType>
</xs:element>
```

2.3 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].
<dss:Base64Signature> [Optional]

A base64 encoding of some non-XML signature, such as a PGP or PKCS7 signature. The type of signature is specified by its MimeType attribute.

<Sig\turningpage{}\nturePtr> [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:

```xml
<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>
```

2.4 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 must 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 will reject the request outright.
The <Outputs> element contains additional protocol outputs. The client selects which outputs the server should respond with by sending certain options. If a client sends no options, he will receive no additional outputs.

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

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

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

2.5 Element <Status>

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

- <StatusCode> [Required]

  A code representing the status of the corresponding request.

- <StatusMessage> [Optional]

  A message which MAY be returned to an operator.

The following schema fragment defines the <Status> element:

```xml
<xs:element name="Status"  
  <xs:complexType>
    <xs:sequence>
      <xs:element name="dss:StatusCode" type="xs:QName"/>
      <xs:element name="StatusMessage" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The <StatusCode> contain an XML Schema QName. A namespace prefix MUST be provided. The QName may be associated with the DSS protocol namespace, or with the namespace of some profile of DSS. Here we define certain generic QNames which are common to all of the DSS protocols:

- Success
  The protocol executed successfully.
- RequestDenied
  The client isn’t authorized to perform the requested operation.
MalformedRequest
   The request didn’t parse correctly or was otherwise invalid.
UnknownOption
   The server didn’t recognize one of the requested options.
InternalError
   The server failed for some reason that is not the client’s fault.
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:

- <Options> [Optional]
  - This element contains all options about the request.
- <InputDocuments> [Required]
  - A list of input documents which the signature will be calculated over.

The following schema fragment defines the <SignRequest> element:

```xml
<xsd:element name="SignRequest">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="dss:Options" minOccurs="0"/>
      <xsd:element ref="dss:InputDocuments"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

3.2 Element <SignResponse>

The <SignResponse> element contains:

- <Status> [Required]
  - A code representing the status of the request.
- <Signature> [Optional]
  - The resultant signature, if the request succeeds.
- <Outputs> [Optional]
  - Any outputs that were requested by the presence of a corresponding option.

The following schema fragment defines the <SignResponse> element:

```xml
<xsd:element name="SignResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="dss:Status"/>
      <xsd:element ref="dss:Signature" minOccurs="0"/>
      <xsd:element ref="dss:Outputs" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```
3.3 Basic Processing

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

- The server hashes each input <Document> or <DocumentURI>.
- The server forms a <ds:Reference> for each Input Document out of its RefURI, RefType, <ds:Transforms>, and hash value.
- The server forms a <ds:SignedInfo> out of the <ds:Reference> elements.
- The server forms a <ds:Signature> by signing the <ds:SignedInfo>.
- 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

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

3.4.1 Element <RequestID>

The <RequestID> element is used to correlate requests with responses. When present in a request, the server will return it in the response <Outputs>.

The following schema fragment defines the <RequestID> element:

```
<xs:element name="RequestID" type="xs:string"/>
```

3.4.2 Element <SignedReferences>

The <SignedReferences> element gives the client greater control over how the <ds:Reference> 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:

- WhichInputDocument [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>
```

### 3.4.3 Element `<ApplicationProfile>`

The `<ApplicationProfile>` element indicates a particular application 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 `<ApplicationProfile>` element:

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

### 3.4.4 Element `<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 his processing.

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

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

### 3.4.5 Elements `<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"/>
```

### 3.4.6 Element `<IntendedAudience>`

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

The following schema fragment defines the `<IntendedAudience>` element:
3.4.7 Element `<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>
```
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:

- <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 list of input documents which the signature was calculated over.

The following schema fragment defines the <VerifyRequest> element:

```xml
<xs:element name="VerifyRequest">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="dss:Options" minOccurs="0"/>
      <xs:element ref="dss:Signature" minOccurs="0"/>
      <xs:element ref="dss:InputDocuments"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

4.2 Element <VerifyResponse>

The <VerifyResponse> element contains:

- <Status> [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.

The following schema fragment defines the <VerifyResponse> element:

```xml
<xs:element name="VerifyResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="dss:Status"/>
      <xs:element ref="dss:Outputs" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```
5 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.

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.
6 References

6.1 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>
</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.