Signature Gateway Profile of the OASIS Digital Signature Service Version 1.0

OASIS Standard

11 April 2007

Specification URIs:

This Version:

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-SignatureGateway-spec-v1.0-os.html

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-SignatureGateway-spec-v1.0-os.pdf

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-SignatureGateway-spec-v1.0-os.doc

Latest Version:

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-SignatureGateway-spec-v1.0-os.html

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-SignatureGateway-spec-v1.0-os.pdf

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-SignatureGateway-spec-v1.0-os.doc

Technical Committee:

OASIS Digital Signature Services TC

Chair(s):

Nick Pope, Thales eSecurity
Juan Carlos Cruellas, Centre d'aplicacions avançades d’Internet (UPC)

Editor:

Glenn Benson, JPMorgan

Related work:

This specification is related to:

- oasis-dss-core-spec-v1.0-os

Abstract:

This document profiles the OASIS DSS core protocol for signature gateway transformation processing. This profile is intended to be generic, so it may be combined with other profiles freely.

Status:

This document was last revised or approved by the membership of OASIS on the above date. The level of approval is also listed above. Check the current location noted above for possible later revisions of this document. This document is updated periodically on no particular schedule.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/dss/.

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 Technical Committee web page (http://www.oasis-open.org/committees/dss/ipr.php.

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/dss/.

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.

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.

The names "OASIS" are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Table of Contents

1 Introduction

1.1 Profile Type

An OASIS DSS profile has exactly one class: *concrete* or *abstract*. The most significant difference between the two classes is that one may directly implement a concrete protocol; however, one may not claim conformance of a specific realization to an abstract protocol. A concrete profile sufficiently constrains the flexibility of the DSS core protocol **[DSSCore]** so that a profile-compliant client and server should be interoperable at the levels of the protocol as defined in the profile. An abstract profile requires further definition of a subordinate concrete profile before an implementer may create a conformant realization.

This document identifies one abstract profile and two concrete profiles. The abstract profile defines all definitions required for DSS interoperability with one exception: transmission binding.

The concrete profiles fill the gap by permitting an implementer to build a realization and claim Signature Gateway Profile realization by both conforming to the abstract profile, and conforming to a permissible transmission binding as defined in one of the concrete profiles.

The two concrete profiles identified in this document each a specific transmission binding:

- HTTP POST Transport Binding, or
- SOAP 1.2 Transport Binding.

The addition of security to these bindings is optional.

Subsequent revisions may either add new concrete profiles in separate documents, or as modifications to this document.

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

1.2 Overview (Non-Normative)

This document standardizes a Signature Gateway by profiling the DSS signing and verifying protocols **[DSSCore]**. This Signature Gateway transforms both *signing technology* and *credential logistics*. The signing technology specifies the mechanisms through which one creates and verifies a signature. Example technologies include, but are not limited to photocopied signatures, Public Key Infrastructure signatures, and signatures defined using symmetric keying material (see **[XMLDSIG]** for some symmetric specifications). Credential logistics, describes the means to distribute credentials to remote parties; and the associated vehicle for distributing trust. Although electronic means allows communication at a distance, geographic separation increases the difficulty of trusting one's peers. Credentials overcome many of the geographic impediments to trust; and the associated logistics securely define the means of managing the credential lifecycle, e.g., distribution, revocation, renewal, and retirement.

Each kind of technology and logistics has its own distinct advantages and disadvantages. As a result, no universal best-of-breed solution exists for all deployment scenarios. Some scenarios require different solutions for distinct spaces; and a gateway serves as an intermediary connector. The DSS Signature Gateway operates in the following use case. A signer applies its signing credential to create a signature. The signer does not transmit the signature directly to a recipient, because the recipient might not understand the signer's signature technology; and the recipient may not trust the signer's credential. Instead, the signer sends the signature to a mutually trusted Signature Gateway which transforms the signature into a format that the recipient validates. The Gateway's transformation operation first validates the original signature, and then creates a new signature. Consider the following example. An organization may allow its employees and machines to trust communication that originates from within the security perimeter, while requiring extra security for externally-originated messages. Rather than distribute the means for secure interoperability throughout the enterprise and extranet, the organization may establish a trusted Signature Gateway. The Gateway validates its incoming messages from the external parties; and then marks the Gateway's stamp of approval which downstream servers consume.

The signature gateway profile may operate in multiple different deployment models. Two example models are described below.

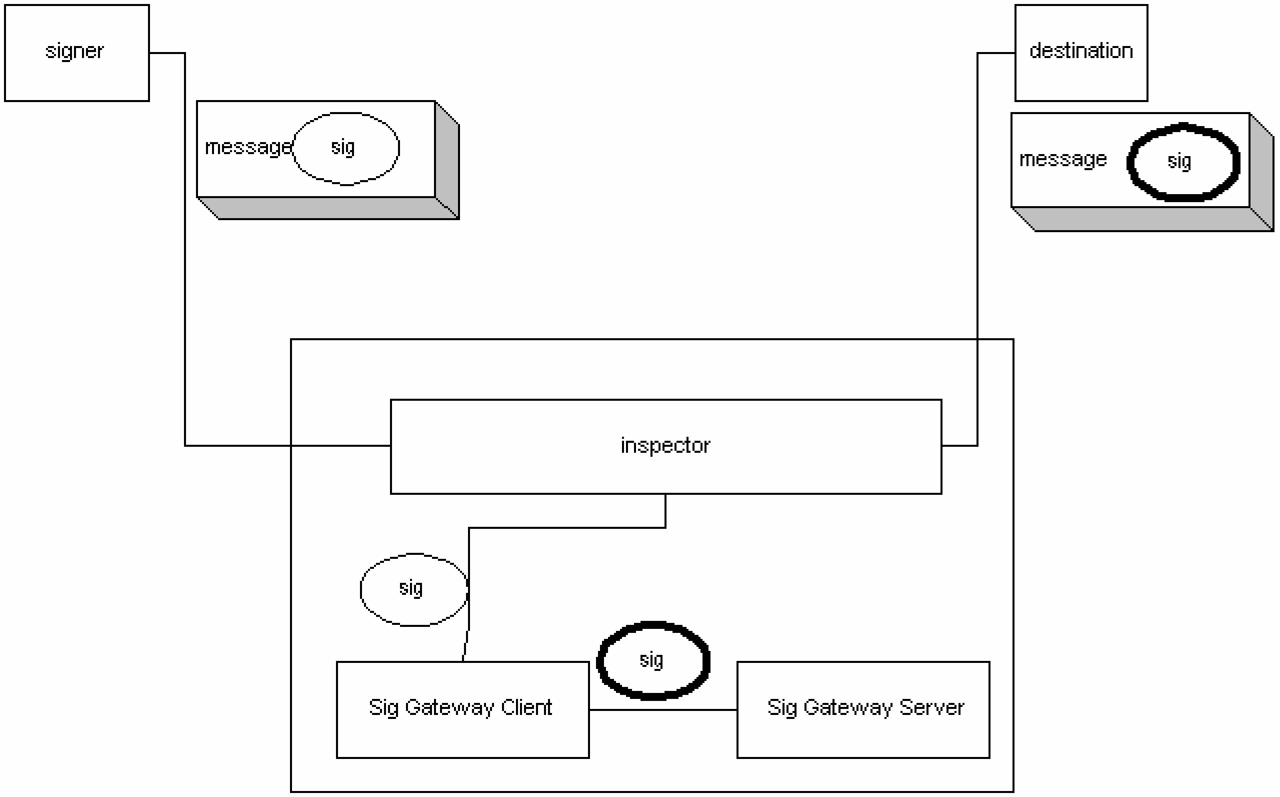
1.3 Request-Response Deployment Model

The request-response deployment model has three actors: signature client, DSS client, and DSS Signature Gateway Server.

1. The signature client signs a document or transaction, and sends the signed data to the DSS client.
2. The DSS client wraps the signed data in the context of DSS Signature Gateway Profile VerifyRequest, and sends the request to the DSS Signature Gateway Server.
3. The DSS Signature Gateway server performs the necessary validation services, and returns a DSS Signature Gateway VerifyResponse to the DSS client.

1.4 In-Line Deployment Model

Devices located at the security perimeter may combine Signature Gateway with other security services. Consider for example, deep packet inspection firewalls, content-inspecting load balancers, intelligent reverse proxies, or XML firewalls. These devices contain the technology to inspect incoming communication while searching for signatures. When the device identifies a signature within the context of a message, the device applies the Signature Gateway transformation, and then forwards the modified communication to the destination. The Figure below illustrates the constituent components:



The request-response deployment model has three actors: signer, inline proxy, and destination. The inline proxy has three constituent components: inspector, Signature Gateway Client, and Signature Gateway Server.

1. The signer sends a message that contains a signature to the in-line proxy.
2. The inspector component of the in-line proxy captures the message and searches for signed data. If the inspector identifies signed data, then the inspector passes the signed data to the DSS Signature Gateway Client.
3. The DSS Signature Gateway Client creates DSS Signature Gateway VerifyRequest using the signed data. The DSS client sends this VerifyRequest to the DSS Signature Gateway Server component.
4. The DSS Signature Gateway Server responds issuing a VerifyResponse.
5. The DSS client passes the response to the inspector component.
6. The inspector modifies the message per the response returned from the DSS Signature Gateway Server and sends the modified message to a downstream, destination application.

1.5 Terminology

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 **[RFC 2119]**. These keywords are capitalized when used to unambiguously specify requirements over protocol features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

This specification uses the following typographical conventions in text: `<ns:Element>`, `Attribute`, **Datatype**, `OtherCode`.

1.6 Namespaces

Conventional XML namespace prefixes are used in this document:

- The prefix `dss:` (or no prefix) stands for the DSS core namespace **[Core-XSD]**.
- The prefix `ds:` stands for the W3C XML Signature namespace **[XMLDSIG]**.

Applications MAY use different namespace prefixes, and MAY use whatever namespace defaulting/scoping conventions they desire, as long as they are compliant with the Namespaces in XML specification **[XML-ns]**.

1.7 Normative References

[Core-XSD] S. Drees et al. *DSS Schema*. OASIS, February 2007

[DSSCore] S. Drees et al. *Digital Signature Service Core Protocols and Elements*. OASIS, February 2007

[DSS-XAdES] Juan Carlos Cruellas et al. XAdES Profile of the OASIS Digital Signature Service

[RFC 2119] S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. IETF RFC 2396, August 1998.

http://www.ietf.org/rfc/rfc2396.txt.

[RFC3369] R. Housley. *Cryptographic Message Syntax*. IETF RFC 3369, August 2002.

http://www.ietf.org/rfc/rfc2459.txt.

[XAdES] XML Advanced Electronic Signatures ETSI TS 101 903, February 2002 *(shortly to be re-issued)* http://pda.etsi.org/pda/home.asp?wki_id=1UFEyx7ORuBCDGED3IiJH

[XML-ns] T. Bray, D. Hollander, A. Layman. *Namespaces in XML.* W3C Recommendation, January 1999.

http://www.w3.org/TR/1999/REC-xml-names-19990114

[XMLDSIG] D. Eastlake et al. *XML-Signature Syntax and Processing.* W3C Recommendation, February 2002.

http://www.w3.org/TR/1999/REC-xml-names-19990114

2 Profile Features

2.1 Identifier

urn:oasis:names:tc:dss:1.0:profiles:siggty

This identifier names an abstract profile. An <AdditionalProfile> identifier is mandatory in order to name a subordinate concrete profile.

2.1.1 Core HTTP Transport Binding

The following `<AdditionalProfile> specifies a concrete profile:`

urn:oasis:names:tc:dss:1.0:HTTP-POST-Transport-binding

This concrete profile requires:

- ingress: HTTP POST Transport binding as specified in the 1.0 core
- egress: unspecified

2.1.2 Core SOAP 1.2 Transport Binding

The following `<AdditionalProfile> specifies a concrete profile:`

urn:oasis:names:tc:dss:1.0:SOAP-Transport-binding

This concrete profile requires:

- ingress: SOAP 1.2 Transport binding as specified in the 1.0 core
- egress: unspecified

2.1.3 Other Transport Bindings Defined as Concrete Sub-Profiles

If the transport binding is defined as in a subordinate profile, then add the requisite identifier as an `<AdditionalProfile>.`

2.2 Scope

This document profiles the DSS signing and verifying protocols defined in **[DSSCore]** and profiles XML signature format for a signature gateway. This document permits other signature formats such as CMS **[RFC3369]**.

2.3 Relationship To Other Profiles

This profile is based directly on the **[DSSCore]**.

This document contains an abstract profile and two concrete protocols.

2.4 Signature Object

This profile supports the verification of incoming signatures and the production of a resultant signature by the gateway. The profile MUST support XMLDSIG **[XMLDSIG]** for both incoming and produced signatures. Other formats are optional. This means that a Signature Gateway MAY accept incoming signatures in a non-XMLDSIG compliant format, e.g., CMS **[RFC3369]**.

2.5 Transport Binding

The combination of this abstract profile and a permissible transport binding provides sufficient specification for interoperability. For the transport bindings see the concrete protocols: **[DSSCore]** HTTP POST Transport binding as named by urn:oasis:names:tc:dss:1.0:HTTP-POST-Transport-binding, and **[DSSCore]** SOAP Transport Binding as named by urn:oasis:names:tc:dss:1.0:SOAP-Transport-binding.

Other permissible transport bindings may be defined in subordinate concrete profiles.

2.6 Security Binding

A security binding is permissible but not required. If used, this profile does not specify or constrain the security binding.

3 Profile of Signing Protocol

3.1 Element <SignRequest>

The <dss:SignRequest> is not supported in the Signature Gateway Profile.

3.2 Element <SignResponse>

The <dss:SignResponse> is not supported in the Signature Gateway Profile.

4 Profile of Verifying Protocol

4.1 Element VerifyRequest

4.2 Element OptionalInputs

The Signature Gateway Profile MAY support any client or server optional input defined in **[DSSCore]**. However, some optional inputs are mandatory, or further clarified as described below.

4.2.1.1 Optional input < ServicePolicy >

The Signature Gateway MUST support the optional input defined in **[DSSCore]** `<dss:ServicePolicy>`. The `<dss:ServicePolicy>` MUST include a description of the signature that the Signature Gateway accepts (ingress). In addition `<dss:ServicePolicy>` MUST either include a description of the signature that the Signature Gateway produces (egress), or explicitly note the policy for the egress signature using the term “unspecified”.

The `<dss:ServicePolicy>` specification for the ingress signature MUST include the following items:

- The type of employed signature: **[XMLDSIG]** or **[RFC3369]**.
- Signature algorithm

The `<dss:ServicePolicy>` specification MAY include additional items such as signature attributes, properties, or policies. Topics include, but are not limited to the items on the following list:

- *Signed References and Properties:* Policy that determines if all the Signature Gateway validates some, or all of the signed references and properties such as the manifest, and timestamp.
- *Revocation:* Policy that specifies the rules by which the Signature Gateway checks revocation on the input signature
- *Signature Coverage:* Policy that determines if the Gateway’s signature covers the original document, the signature, the manifest, the signature properties, or some combination of the above.
- *Timestamp:* Policy that specifies any requirement for a timestamp, including the format.
- *Revocation:* Policy that specifies the format, and server that provides revocation information.

A Signature Gateway server MUST support at least one Service Policy. In the Signature Gateway Profile, the `<dss:ServicePolicy>` is NOT optional, i.e., the client must provide it in each request. A Signature Gateway MAY publish its service policy, where the means for publication is outside the scope of DSS.

4.2.1.2 OptionalInput < ReturnUpdatedSignature >

Each <dss:`VerifyRequest> MUST contain` the optional input defined in**[DSSCore]** `<dss:ReturnUpdatedSignature>`. The DSS Server MUST NOT sign the input document unless it first validates the input `<dss:SignatureObject>` successfully.

4.3 Element <VerifyResponse>

4.3.1 Element <ResultMajor>

If the <dss:VerifyRequest> misses any of the required <dss:OptionalInputs>, then the DSS server MUST return the following response in <dss:ResultMajor>.

```
urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError
```

4.3.2 Element <ResultMinor>

If the <dss:VerifyRequest> misses any of the required <dss:OptionalInputs>, then the DSS server MUST return the following response in <dss:ResultMinor>:

```
urn:oasis:names:tc:dss:1.0:resultminor:siggty:NotSupported
```

```
The <dss:ResultMessage> SHOULD contain the identity of the missing
required <dss:OptionalInputs>.
```

4.3.2.1 Signature type mismatch with requested key

If the <dss:VerifyRequest> explicitly specifies a <dss:KeySelector>, where the Signature Gateway's key is not valid, then the Signature Gateway MUST return an error with the following code in <dss:ResultMinor>:

```
urn:oasis:names:tc:dss:1.0:resultminor:siggty:KeyNotSupported
```

4.3.2.2 Signature policy not supported

If the <dss:VerifyRequest> explicitly specifies an unsupported <dss:ServicePolicy>, then the Signature Gateway MUST return an error with the following code in <dss:ResultMinor>.

```
urn:oasis:names:tc:dss:1.0:resultminor:siggty:ServicePolicyNotSupported
```

4.3.3 Element <OptionalOutputs>

4.3.3.1 OptionalOutput < UpdatedSignature >

If the Signature Gateway Server fails to validate the signature in the VerifyRequest, then the Signature Gateway Server MUST NOT include the <dss:UpdatedSignature>. If the Signature Gateway Server successfully validates the signature in the VerifyRequest, then the Signature Gateway Server SHOULD include the <dss:UpdatedSignature>

5 Profile of Signatures

The profile MAY support the XML Signature as defined in **[XMLDSIG]** or **[XAdES]**. within the `<ds:object>` element of the XML signature.

The profile MAY support the CMS signature as defined in **[RFC3369]** specified as a `<Base64Signature>` as defined in **[DSSCore]**.

6 Server Processing Rules

6.1 VerifyRequest

In addition to the processing specified in **[DSSCore]**, the DSS server additionally validates the existence of all required optional inputs. The DSS server MUST NOT produce a signature unless it first successfully validates the client’s signature in accordance with the Service Policy.

A. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Burt Kaliski, RSA Security
John Linn, RSA Security
Trevor Perrin, Individual