Abstract Code-Signing Profile of the OASIS Digital Signature Services

Committee Draft , 24 December 2004 (Working Draft 05)

Document identifier:

oasis-dss-1.0-profiles-codesigning-spec-cd-01

Location:

http://docs.oasis-open.org/dss/

Editor:

Pieter Kasselman, Cybertrust <pieter.kasselman@cybertrust.com>

Contributors:

Trevor Perrin

Abstract:

This draft profiles the OASIS DSS core protocols and the Asynchronous Processing Abstract Profile of the OASIS Digital Signature Services for the purpose of creating code-signing signatures.

Status:

This is a **Committee 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

The DSS signing and verifying protocols are defined in **[DSS Core]** and asynchronous processing for DSS messages are defined in **[DSS Async]**. As defined in those documents, these protocols have a fair degree of flexibility and extensibility. This is an abstract profile of **[DSS Core]** and **[DSS Async]**. It also profiles the processing rules followed by clients and servers when using these protocols.

The resulting profile is an *abstract profile*. Further profiles will build on this one to provide a basis for implementation and interoperability.

The following sections provide guidance to interpreting the rest of this document.

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 **[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`.

1.2 Namespaces

The structures described in this specification are contained in the schema file **[CS-XSD]**. All schema listings in the current document are excerpts from the schema file. In the case of a disagreement between the schema file and this document, the schema file takes precedence.

This schema is associated with the following XML namespace:

```
urn:oasis:names:tc:dss:1.0:profiles:codesigning:1.0
```

If a future version of this specification is needed, it will use a different namespace.

Conventional XML namespace prefixes are used in this document:

- The prefix `dsscs:` (or no prefix) stands for the DSS code-signing namespace **[CS-XSD]**.
- The prefix `dss:` stands for the DSS core namespace **[Core-XSD]**.
- The prefix `async:` stands for this profiles namespace **[Async-XSD]**.
- The prefix `ds:` stands for the W3C XML Signature namespace **[XMLSig]**.

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]**.

1.3 Overview (Non-normative)

The DSS signing and verifying protocols are defined in **[DSS Core]**. Asynchronous processing of DSS signing and verification protocols are defined in **[DSS Async]**. As defined in that document, these protocols have a fair degree of flexibility and extensibility.

This specification provides an abstract profile of the DSS signing messages for the case where the object or input document that is being signed is a software program that can be executed on a computing platform. The software program may be in source form, or in compiled form. The process for signing these software programs is referred to as code-signing. Code-signing allows the recipient of a software program to receive assurances regarding the origin and integrity of the program. The recipient may use this information to make a trust decision on whether to install or execute a software program.

Traditionally the task of generating the signature on the software program is left to the software developer. However it may not always be appropriate to combine the roles of the software developer and the code-signer. By centralizing the generation of signatures in the code-signing process, the role of the software developer and the code signer is easily separated. This has the advantage that keys used for signing software programs can be better managed, access to the keys can be better controlled, audit trails can be centrally kept, event records can be reliably archived and signing policies can be rigorously enforced.

In the centralized code-signing model, the software developer is responsible for the creation and development of a program. The software developer may also perform some basic testing of the software program. Before distributing the software program, the software developer may need to have the software program digitally signed to convey assurances regarding the origin and authenticity of the software program to the receiving platform or user. In order to obtain a signature the software developer contacts the code-signing service and requests a signature for the software program. Part of this request may include authentication information to allow the code-signing service to make a decision on whether the software developer is authorized to request a signature for the software program. The request may also include the software program in source form, compiled form or both. The centralized code-signing server may then generate a signature. Generation of the signature may be subject to numerous criteria, including whether the software developer is authorized to request the signature and whether the software program conforms to the norms and standards set by the code-signing service. The code-signing service may decline to generate a signature if all of its criteria and conditions are not met. The exact criteria for generating a signature are subject to the policies of the code-signing service and are beyond the scope of this document and may be further specified as part of a concrete profile. Once the signature is generated, the result is returned to the software developer and the signed software program may be distributed.

Depending on the policies and criteria of the code-signing service, there may be a substantial delay between the time of submission of the software program and the time of signature generation. This delay may make synchronous message exchange impractical and necessitate the use of asynchronous message exchange. The use of asynchronous message exchange allows the software developer to submit the request to the code-signing service, without receiving an immediate response containing the signature. The code signing service responds by acknowledging the receipt of the request. The software developer may then periodically poll the code-signing service to retrieve the signed software program, or may retrieve the signed software program once it receives a notification from the code-signing server.

This asynchronous behavior can be achieved by combining the **[DSS Core]** with the **[DSS Async]** profile. The object for which the signature is requested is included under `<InputDocuments>`. The server may respond synchronously with a `<dss:SignResponse>`

message. If the server can not fulfill the code-signing request synchronously, it responds with a <dss:ResultMajor> code indicating that the request is pending and the <dss:SignatureObject> element is left undefined, as specified in **[DSS Async].** If the signature request is processed asynchronously, the client may request the signature from the server using the <async:PendingRequest> message. The client may poll the server periodically by sending this message, or it may send the message in response to a notification received. The server may respond to the <async:PendingRequest> message by either indicating that the request is still pending or it may return the signature or signed software program. Either of these will be part of the <dss:SignResponse> message.

This document profiles and extends the **[DSS Core]** and **[DSS Async]** specifications to enable the code-signing scenarios sketched.

This document does not provide a profile of the DSS verification messages and does not specify a notification mechanism.

2 Profile Features

2.1 Identifier

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

A server implementing this profile MAY support asynchronous processing as defined in the asynchronous processing profile as defined in **[DSS Async].**

The client MUST implement asynchronous processing as defined in **[DSS Async]** and MUST include the **urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing** identifier in the `<dss:AdditionalProfile>` element.

2.2 Scope

This document profiles the DSS signing protocol and Asynchronous Processing Protocol as defined in **[DSS Core]** and **[DSS Async]**.

2.3 Relationship To Other Profiles

This profile is based directly on the **[DSS Core]** and **[DSS Async].**

This profile is an abstract profile which can not be implemented directly, and may be further profiled.

2.4 Signature Object

This profile is intended to provide a general framework for code-signing signature services and does not specify or constrain the type of signature object. It is up to future profiles of this abstract profile to constrain the type of signature object.

2.5 Transport Binding

This profile does not specify or constrain the transport binding.

2.6 Security Binding

This profile does not specify or constrain the security binding.

3 Profile of Signing Protocol

3.1 Element <dss:SignRequest>

3.1.1 Element <dss:OptionalInputs>

None of the optional inputs specified in the **[DSS Core]** are precluded in this abstract profile.

The `<AdditionalProfile>` element MUST be present, and MUST include the **urn:oasis:names:tc:dss:1.0:profiles:asynchronousprocessing** identifier.

3.1.2 Element <dss:InputDocuments>

This is an abstract profile and no constraints are imposed on the type of input document for which a signature may be requested.

3.2 Element <dss:SignResponse>

3.2.1 Element <dss:Result>

This profile defines no additional `<dss:ResultMinor>` codes.

3.2.2 Element <dss:OptionalOutputs>

None of the optional outputs specified in the **[DSS Core]** are precluded in this abstract profile.

3.2.3 Element <dss:SignatureObject>

This is an abstract profile and no constraints are imposed on the signature type that may be returned. If a signature or signed software program is returned, it MUST be included under this element.

4 Profile of Verifying Protocol

This document does not provide a profile of the DSS verification messages.

5 Profile of Code-signing Signatures

This is an abstract profile and no constraints are imposed on the type of signatures that are allowed.

6 Profile of Server Processing Rules

A DSS server that performs code-signing SHOULD perform the following steps upon receiving a <dss:SignRequest>.

- Determine if the requesting user is authorized to submit a <dss:SignRequest> or <async:PendingRequest> message for the purpose of code-signing request. This decision may be based on the authentication provided by the transport and security bindings.
- If a <dss:SignRequest> message is received, the server may respond synchronously by generating the signature according to the concrete code-signing profile and include the requested signature in the <dss:SignResponse> message. Alternatively the server may respond with a <dss:SignResponse> message containing a <dss:ResultMajor> error code of Pending, indicating asynchronous processing of the request.. The signature may then be generated at the convenience of the server. The requested signature may then be retrieved by the client using subsequent <async:PendingRequest> messages.
- If the <async:PendingRequest> message is used to retrieve a signature on a previously submitted software program, the server may use the OriginalRequestId attribute to determine if the requested signature has been generated.
    - If the signature is available, the server SHOULD respond with a <dss:SignResponse> message containing the requested signature.
    - If the signature request has not been processed, the server SHOULD respond with a <dss:SignResponse> message containing a <ResultMajor> error code of Pending. The client SHOULD submit a <async:PendingRequest> at a later time to retrieve the signature.
    - If the signature could not be generated the server SHOULD respond with a <dss:SignResponse> message containing a <ResultMajor> error code indicating the reason why the signature could not be generated.

7 Profile of Client Processing Rules

A DSS client that requests signatures from a DSS code-signing server SHOULD perform the following steps.

- The client MUST support asynchronous processing as defined in **[DSS Async]**.
- The client SHOULD authenticate itself using one of the mechanisms available through the transport or security bindings.
- If this is a new signature request the client MUST submit an original signature request for a software program using the `<dss:SignRequest>` message.
    - If the client receives a `<dss:SignResponse>` message containing the requested signature, the code-signing process is complete
    - If the client receives a `<dss:SignResponse>` message containing a `<ResultMajor>` element indicating that the request is pending, the client retains the `RequestID` attribute and MAY submit a `<async:PendingRequest>` message which MUST include the `OriginalRequestID` attribute. The value of the `OriginalRequestID` attribute MUST be set to the value of the `RequestID` attribute used in the initial signature request.
- If this is a signature retrieval request (i.e. a request to retrieve a signature that was previously requested but not returned) the client MUST use the `<async:PendingRequest>` message which MUST include the `OriginalRequestID` attribute. The value of the `OriginalRequestID` attribute MUST be set to the value of the `RequestID` attribute used in the initial signature request. The client may be required to provide authentication information through a mechanism defined through the transport or security binding.
    - If the client receives a `<dss:SignResponse>` message containing the requested signature, or an error code other than `Pending,` the code-signing process is complete.
    - If the client receives a `<dss:SignResponse>` message containing a `<dss:RequestMajor>` element indicating that the request is still pending, the client MAY re-submit a `<async:PendingRequest>` message at a later time. The `<async:PendingRequest>` message MUST include the `OriginalRequestID` attribute. The value of the `OriginalRequestID` attribute MUST be set to the value of the `RequestID` attribute used in the initial signature request.

8 Editorial Issues

1) *Updated from version one. Removed code-signing specific async mechanism in favour of the async profile specified in [DSS Async].*
2) *Updated from version 2 to use version 4 of async processing profile.*

9 References

9.1 Normative

[Async-XSD] A, Kuehne. *Asynchronous Processing Profile Schema*. OASIS, **(MONTH/YEAR TBD)**

[CS-XSD] P.Kasselman, *DSS Abstract Code-Signing Schema*. OASIS, **(MONTH/YEAR TBD)**

[Core-XSD] T. Perrin et al. *DSS Schema*. OASIS, **(MONTH/YEAR TBD)**

[DSS Core] T. Perrin et al. *Digital Signature Service Core Protocols and Elements*. OASIS, **(MONTH/YEAR TBD)**

[DSS Async] Asynchronous Processing Abstract Profile of the OASIS Digital Signature Services, Working Draft 04, 21 August 2004

[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.

[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

[XMLSig] D. Eastlake et al. *XML-Signature Syntax and Processing.* W3C Recommendation, February 2002. http://www.w3.org/TR/1999/REC-xml-names-19990114

Appendix A. Revision History

Rev	Date	By Whom	What
wd-01	2004-01-08	Pieter Kasselman	Initial version based oasis-dss-1.0-profiles-XYZ-spec-wd-03.doc by Trevor Perrin
wd-02	2004-06-25	Pieter Kasselman	New version based on oasis-dss-1.0-profiles-XYZ-spec-wd-04.doc. Remove code-signing specific async capabilities in favor of [DSS Async] mechanisms.
wd-03	2004-10-13	Pieter Kasselman	Editorial corrections and updates to take into account version four of [DSS Async].
wd-04	2004-11-24	Pieter Kasselman	Editorial corrections based on feedback from Trevor Perrin
wd-05	2004-11-26	Pieter Kasselman	Removed reference to `<ResponseMechanism>` to reflect changes in version wd-05 of async profile
cd-01	2004-12-24	Pieter Kasselman	Approved Committee Draft

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.