J2ME Code-Signing Profile of the OASIS Digital Signature Services

Committee Specification

13 February 2007

This Version:

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-codesigning-j2me-spec-cs-v0.1-r1.html

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-codesigning-j2me-spec-cs-v0.1-r1.pdf

Latest Version:

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-codesigning-j2me-spec-cs-v0.1-r1.html

http://docs.oasis-open.org/dss/v1.0/oasis-dss-profiles-codesigning-j2me-spec-cs-v0.1-r1.pdf

Technical Committee:

OASIS Digital Signature Services TC

Chair(s):

Nick Pope, Thales eSecurity

Juan Carlos Cruellas, Centre d'aplicacions avançades d'Internet (UPC)

http://docs.oasis-open.org/dss/v1.0/

Editor:

Andreas Kuehne, *individual*

Related work:

This specification is related to:

- oasis-dss-core-spec-cs-v1.0-r1

Abstract:

This document profiles the OASIS DSS core protocols and the OASIS DSS Abstract Code-Signing Profile for the purpose of creating J2ME code-signing signatures.

Status:

This document was last revised or approved by the OASIS Digital Signature Services TC 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

Table of Contents

1 Introduction

The DSS signing and verifying protocols are defined in **[DSS Core]** and the code-signing profile of the DSS signing and verification protocols are defined in **[DSS CS].** As defined in those documents, these protocols have a fair degree of flexibility and extensibility. This document profiles these protocols to limit their flexibility and extend them in concrete ways. It also profiles the processing rules followed by clients and servers when using these protocols, and profiles the J2ME signature format for use with these protocols. The resulting profile is suitable for implementation and interoperability.

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

1.1 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`.

1.2 Namespaces

The structures described in this specification are contained in the schema file **[J2ME-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:J2ME: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 `dsscsj2me:` (or no prefix) stands for the DSS code-signing namespace **[CS-XSD]**.
- The prefix `dsscs:` stands for the DSS code-signing namespace **[CS-XSD]**.
- The prefix `async:` stands for this profiles namespace **[Async-XSD]**.
- The prefix `dss:` stands for the DSS core namespace **[Core-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 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

[RFC2119]	S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. IETF RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.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
[DSS CS]	Abstract Code-Signing Profile of the OASIS Digital Signature Services Working Draft 03, 13 October 2004
[DSS Async]	Asynchronous Processing Abstract Profile of the OASIS Digital Signature Services, Working Draft 04, 21 August 2004
[CS-XSD]	P. Kasselman, *Codesigning Schema*. OASIS, **(MONTH/YEAR TBD)**
[Async-XSD]	A, Kuehne. *Asynchronous Processing Profile Schema*. OASIS, **(MONTH/YEAR TBD)**
[J2ME-CS-XSD]	P. Kasselman, *J2ME Codesigning Schema*. OASIS, **(MONTH/YEAR TBD)**
[MIDP 2.0]	Mobile Information Device Profile for Java™ 2 Micro Edition Version 2.0, JSR 118 Expert Group
[RFC 2437]	RFC 2437 PKCS #1: RSA Cryptography Specifications Version 2.0, B. Kaliski, J. Staddon, http://www.ietf.org/rfc/rfc2437.txt

1.4 Overview (Non-normative)

The **[DSS-CS]** abstract profile provides a profile of **[DSS-Core]** and combines it with the **[DSS-Async]** profile. The **[DSS-CS]** profile allow for the generation of signatures on content, including software programs, and is flexible enough to accommodate the typical scenarios encountered in the software development lifecycle.

This specification provides a concrete profile based on **[DSS-CS]** for requesting the generation of signatures as specified in the Java 2 Micro Edition (J2ME), Mobile Information Device Profile 2.0 **[MIDP 2.0]**.

2 Profile Features

2.1 Identifier

urn:oasis:names:tc:dss:1.0:profiles:codesigning:1.0:J2ME:1.0

2.2 Scope

This document further profiles the abstract profile for code-signing as described in **[DSS CS]**, which is a profile of the DSS signing protocol defined in **[DSS Core]** in combination with **[DSS Async]**.

2.3 Relationship To Other Profiles

This profile is a concrete profile of the abstract code-signing profile defined in **[DSS CS]**.

2.4 Signature Object

This profile supports the creation of signatures as defined in **[MIDP 2.0]**. **[MIDP 2.0]** defines the use of EMSA-PKCS1-v1_5 as defined in **[RFC 2437]**.

2.5 Transport Binding

This profile is transported using the HTTP POST Transport Binding defined in **[DSS Core]**.

2.6 Security Binding

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

3 Profile of Signing Protocol

3.1 Element <dss:SignRequest>

3.1.1 Element <dss:OptionalInputs>

Optional inputs MUST be used as defined in **[DSS CS]**.

The following optional inputs defined in the **[DSS Core]** will not be understood by a server implementing this profile:

- `<dss:AddTimeStamp>`
- `<dss:SignedReference>`
- `<dss:Properties>`
- `<dss:SignaturePlacement>`
- `<dss:EnvelopingSignature>`

In addition the following constraints are placed on the optional inputs as described below.

3.1.1.1 Element <dss:SignatureType>

The `<dss:SignatureType>` MUST contain the identifier `urn:ietf:rfc:2437:RSASSA-PKCS1-v1_5`. This refers to PKCS #1 version 1.5 signatures as defined in **[RFC 2437]**.

3.1.1.2 Element <dss:ServicePolicy>

The `<dss:ServicePolicy>` SHOULD be used to indicate a specific server signing policy. The server signing policy is mapped to the recommended security policy for GSM/UMTS compliant devices in **[MIDP 2.0].** The following URIs may be used to specify the service policy and corresponding domain under which the MIDlet must be signed.

For code that should execute in the manufacturer domain use:

`urn:oasis:names:tc:dss:1.0:profiles:codesigning:1.0:J2ME:1.0:manufacturer`

For code that should execute in the operator domain use:

`urn:oasis:names:tc:dss:1.0:profiles:codesigning:1.0:J2ME:1.0:operator`

For code that should execute in the trusted third party domain use:

`urn:oasis:names:tc:dss:1.0:profiles:codesigning:1.0:J2ME:1.0:trustedisv`

3.1.2 Element <dss:InputDocuments>

The server MUST accept `<dss:Document>` inputs and MUST NOT accept `<dss:DocumentHash>` inputs. A server that implements this profile MUST respond with a `<dss:ResultMajor>` code of `urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError` as defined in **[DSS Core]** if it receives a `<dss:DocumentHash>` input.

The `<dss:Document>` element MUST include the Base64 encoded J2ME JAR file on which the signature must be calculated within a `<dss:Base64Data>` element. The `MimeType` attribute

MUST be set to `application/java-archive`. Only one `<Document>` element MUST be submitted.

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. In addition this profile defines the following `<dss:OptionalOutputs>`:

- `<X509CertificatePath>`

In addition, the `<dss:OptionalOutputs>` element MAY contain a `<dss:Document>` element.

3.2.2.1 Element <X509CertificatePath>

This element defines the certificate path including the certificate containing the public key required to verify the signature generated on the JAR file submitted by the client and all intermediary certificates, excluding the root certificate. The client MAY use this information to determine the appropriate entries in the Java Application Descriptor file (JAD) file that is distributed with the JAR file containing the MIDP 2.0 application. The server may return multiple `<X509CertificatePath>` elements. The orders of the <X509CertificatePath> elements are significant. The first `<X509CertificatePath>` element corresponds to the first certificate path, identified by n=1 in the JAD file, the second `<X509CertificatePath>` element corresponds to the second certificate path, identified by n=2, in the JAD file, the j'th `<X509CertificatePath>` element corresponds to the j'th certificate path, identified by n=j, in the JAD file. The `<X509CertificatePath>` element contains the following elements:

`<X509Certificate>`

The `<X509Certificate>` element contains a base64-encoded X.509 v3 certificate. The order of the `<X509Certificate>` elements are significant. The first `<X509Certificate>` element contains the signing certificate and corresponds to m=1 in the JAD file for the current `<X509CertificatePath>` element, the second `<X509Certificate>` element contains the first intermediary certificate and corresponds to m=2 the current `<X509CertificatePath>` element, the k'th `<X509Certificate>` element contains the k-1'st intermediary certificate that issued the k-2'nd intermediary cert.

```
<xs:element name="X509CertificatePath"
            type="dsscsj2me:X509CertificatePathType"/>

<xs:complexType name="X509CertificatePathType">
   <xs:sequence maxOccurs="unbounded">
          <xs:element ref="dsscsj2me:X509Certificate"/>
   </xs:sequence>
</xs:complexType>
```

```
<xs:element name="X509Certificate"
            type="dsscsj2me:X509CertificateType"/>

<xs:simpleType name="X509CertificateType">
   <xs:restriction base="xs:base64Binary"/>
</xs:simpleType>
```

3.2.2.2 Element <dss:Documents>

The server MAY include the J2ME JAR file on which the signature was created as an optional output using the `<dss:Documents>` element. If the `<dss:Document>` element is included in the response as an optional output, it MUST include the Base64 encoded J2ME JAR file within a `<dss:Base64Data>` element. The included J2ME JAR file MUST be the file on which the signature included in the `<dss:SignatureObject>` was calculated. The `MimeType` attribute MUST be set to `application/java-archive`.

3.2.3 Element <dss:SignatureObject>

The server MUST return a Base64 encoded PKCS #1 signature within the `<Base64Signature>` element. The `<dss:SignatureObject>` element MUST NOT contain any other elements.

4 Profile of Verifying Protocol

This **[DSS CS]** profile does not provide a profile of the DSS verification messages and consequently a server implementing this profile MUST NOT respond to any `<dss:VerifyRequest>` messages.

5 Profile of J2ME MIDP 2.0 Signatures

The J2ME MIDP 2.0 signature format is fully defined in **[MIDP 2.0]** and no further profiling is required.

6 Profile of Server Processing Rules

The signature must be calculated on the Base64 decoded JAR file. The server processing rules defined in **[DSS CS]** SHOULD be followed.

7 Profile of Client Processing Rules

Client processing rules as defined in **[DSS CS]** SHOULD be followed.

Appendix A. Acknowledgements

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

Participants:

Trevor Perrin, *individual*

Pieter Kasselman, Cybertrust