



Abstract Code-Signing Profile of the OASIS Digital Signature Services

OASIS Standard

11 April 2007

Specification URIs:

This Version:

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

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

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

Latest Version:

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

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

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

Andreas Kuehne, *individual*

Abstract:

This document 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 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

37 the Intellectual Property Rights section of the Technical Committee web page
38 (<http://www.oasis-open.org/committees/dss/ipr.php>.
39 The non-normative errata page for this specification is located at [http://www.oasis-
open.org/committees/dss/](http://www.oasis-
40 open.org/committees/dss/).

41 Notices

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

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

54 Copyright © OASIS® 1993–2007. All Rights Reserved.

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

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

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

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

76 **Table of Contents**

77 1 Introduction 5
78 1.1 Notation 5
79 1.2 Namespaces 5
80 1.3 Normative References 5
81 1.4 Overview (Non-normative) 6
82 2 Profile Features..... 8
83 2.1 Identifier..... 8
84 2.2 Scope 8
85 2.3 Relationship To Other Profiles 8
86 2.4 Signature Object..... 8
87 2.5 Transport Binding..... 8
88 2.6 Security Binding 8
89 3 Profile of Signing Protocol..... 9
90 3.1 Element <dss:SignRequest>..... 9
91 3.1.1 Element <dss:OptionalInputs>..... 9
92 3.1.2 Element <dss:InputDocuments>..... 9
93 3.2 Element <dss:SignResponse> 9
94 3.2.1 Element <dss:Result>..... 9
95 3.2.2 Element <dss:OptionalOutputs>..... 9
96 3.2.3 Element <dss:SignatureObject> 9
97 4 Profile of Verifying Protocol..... 10
98 5 Profile of Code-signing Signatures 11
99 6 Profile of Server Processing Rules 12
100 7 Profile of Client Processing Rules 13
101 Appendix A. Acknowledgements 14

102 1 Introduction

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

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

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

111 1.1 Notation

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

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

120 1.2 Namespaces

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

124 This schema is associated with the following XML namespace:

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

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

127 Conventional XML namespace prefixes are used in this document:

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

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

135 1.3 Normative References

- | | | |
|-----|--------------------|--|
| 136 | [Async-XSD] | A, Kuehne. <i>Asynchronous Processing Profile Schema</i> . OASIS,
137 February 2007 |
| 138 | [CS-XSD] | P.Kasselmann, <i>DSS Abstract Code-Signing Schema</i> . OASIS, February
139 2007 |
| 140 | [Core-XSD] | S Dress et al. <i>DSS Schema</i> . OASIS, February 2007 |

141	[DSS Core]	S Drees et al. <i>Digital Signature Service Core Protocols and Elements</i> . OASIS, February 2007
142		
143	[DSS Async]	A, Kuehne <i>Asynchronous Processing Abstract Profile of the OASIS Digital Signature Services</i> , February 2007
144		
145	[RFC2119]	S. Bradner. <i>Key words for use in RFCs to Indicate Requirement Levels</i> . IETF RFC 2119, March 1997.
146		http://www.ietf.org/rfc/rfc2119.txt .
147		
148	[XML-ns]	T. Bray, D. Hollander, A. Layman. <i>Namespaces in XML</i> . W3C Recommendation, January 1999.
149		http://www.w3.org/TR/1999/REC-xml-names-19990114
150		
151	[XMLSig]	D. Eastlake et al. <i>XML-Signature Syntax and Processing</i> . W3C Recommendation, February 2002.
152		http://www.w3.org/TR/1999/REC-xml-names-19990114
153		
154		

155 **1.4 Overview (Non-normative)**

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

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

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

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

191 Depending on the policies and criteria of the code-signing service, there may be a substantial
 192 delay between the time of submission of the software program and the time of signature

193 generation. This delay may make synchronous message exchange impractical and necessitate
194 the use of asynchronous message exchange. The use of asynchronous message exchange
195 allows the software developer to submit the request to the code-signing service, without receiving
196 an immediate response containing the signature. The code signing service responds by
197 acknowledging the receipt of the request. The software developer may then periodically poll the
198 code-signing service to retrieve the signed software program, or may retrieve the signed software
199 program once it receives a notification from the code-signing server.

200 This asynchronous behavior can be achieved by combining the **[DSS Core]** with the **[DSS**
201 **Async]** profile. The object for which the signature is requested is included under
202 `<InputDocuments>`. The server may respond synchronously with a `<dss:SignResponse>`
203 message. If the server can not fulfill the code-signing request synchronously, it responds with a
204 `<dss:ResultMajor>` code indicating that the request is pending and the
205 `<dss:SignatureObject>` element is left undefined, as specified in **[DSS Async]**. If the
206 signature request is processed asynchronously, the client may request the signature from the
207 server using the `<async:PendingRequest>` message. The client may poll the server
208 periodically by sending this message, or it may send the message in response to a notification
209 received. The server may respond to the `<async:PendingRequest>` message by either
210 indicating that the request is still pending or it may return the signature or signed software
211 program. Either of these will be part of the `<dss:SignResponse>` message.

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

214 This document does not provide a profile of the DSS verification messages and does not specify
215 a notification mechanism. Notification are currently not included within the **[DSS Async]**
216 specification.

217 2 Profile Features

218 2.1 Identifier

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

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

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

225 2.2 Scope

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

228 2.3 Relationship To Other Profiles

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

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

232 2.4 Signature Object

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

236 2.5 Transport Binding

237 This profile does not specify or constrain the transport binding.

238 2.6 Security Binding

239 This profile does not specify or constrain the security binding.

240

241 **3 Profile of Signing Protocol**

242 **3.1 Element <dss:SignRequest>**

243 **3.1.1 Element <dss:OptionalInputs>**

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

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

247 **3.1.2 Element <dss:InputDocuments>**

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

250 **3.2 Element <dss:SignResponse>**

251 **3.2.1 Element <dss:Result>**

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

253 **3.2.2 Element <dss:OptionalOutputs>**

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

255 **3.2.3 Element <dss:SignatureObject>**

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

259

260

261 **4 Profile of Verifying Protocol**

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

263

264 **5 Profile of Code-signing Signatures**

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

267

268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300

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.

301 7 Profile of Client Processing Rules

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

- 304
- 305 • The client MUST support asynchronous processing as defined in **[DSS Async]**.
 - 306 • The client SHOULD authenticate itself using one of the mechanisms available through
307 the transport or security bindings.
 - 308 • If this is a new signature request the client MUST submit an original signature request for
309 a software program using the `<dss:SignRequest>` message.
 - 310 ○ If the client receives a `<dss:SignResponse>` message containing the
311 requested signature, the code-signing process is complete
 - 312 ○ If the client receives a `<dss:SignResponse>` message containing a
313 `<ResultMajor>` element indicating that the request is pending, the client
314 retains the `RequestID` attribute and MAY submit a
315 `<async:PendingRequest>` message which MUST include the
316 `OriginalRequestID` attribute. The value of the `OriginalRequestID`
317 attribute MUST be set to the value of the `RequestID` attribute used in the initial
318 signature request.
 - 319 • If this is a signature retrieval request (i.e. a request to retrieve a signature that was
320 previously requested but not returned) the client MUST use the
321 `<async:PendingRequest>` message which MUST include the `OriginalRequestID`
322 attribute. The value of the `OriginalRequestID` attribute MUST be set to the value of
323 the `RequestID` attribute used in the initial signature request. The client may be required
324 to provide authentication information through a mechanism defined through the transport
325 or security binding.
 - 326 ○ If the client receives a `<dss:SignResponse>` message containing the
327 requested signature, or an error code other than `Pending`, the code-signing
328 process is complete.
 - 329 ○ If the client receives a `<dss:SignResponse>` message containing a
330 `<dss:RequestMajor>` element indicating that the request is still pending, the
331 client MAY re-submit a `<async:PendingRequest>` message at a later time.
332 The `<async:PendingRequest>` message MUST include the
333 `OriginalRequestID` attribute. The value of the `OriginalRequestID`
334 attribute MUST be set to the value of the `RequestID` attribute used in the initial
335 signature request.
- 336

337 **Appendix A. Acknowledgements**

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

340 **Participants:**

341 Trevor Perrin, *individual*

342 Pieter Kasselmann, Cybertrust

343