



1

2 DSS Use Case Requirements 3 Analysis

4 **Working Draft 12, 16 Aug 2003**

5 **Document identifier:**

6 oasis-dss-1.0-requirements-wd-12

7 **Location:**

8 <http://www.oasis-open.org/apps/org/workgroup/dss/>

9 **Editor:**

10 Trevor Perrin <trevp@trevp.net>

11 **Contributors:**

12 John Messing <jmessing@law-on-line.com>

13 Nick Pope <pope@secstan.com>

14 Krishna Sankar" <ksankar@cisco.com>

15 **Abstract:**

16 This document analyses the use cases given in the OASIS DSS Use Case document and
17 identifies the requirements of Digital Signature Services (DSS) protocols.

18 **Status:**

19 This document is a work in progress. Comments are welcome, and should be sent to the
20 dss@lists.oasis-open.org list.

21 Copyright © 2002 and 2003 The Organization for the Advancement of Structured Information
22 Standards [OASIS]

23

23 Table of Contents

24	1	Introduction	4
25	2	Particular Features of Use Cases	4
26	2.1	Corporate Seal	4
27	2.2	SOAP Signing	4
28	2.3	Identified Requestor.....	4
29	2.4	Individual Signatures	4
30	2.5	Long Term Corporate Signatures.....	4
31	2.6	Delegated Signature Verification.....	5
32	2.7	Securing the Transform Chain	5
33	2.8	Non-XML Data Signing	5
34	2.9	ENotarization in presence of notary	5
35	2.10	Court Filings	6
36	2.11	Client-Side Hashing.....	6
37	2.12	Time-stamping / Time-marking.....	6
38	3	Requirements.....	7
39	3.1	Work Products of the TC.....	7
40	3.1.1	Protocols and Core Elements Document.....	7
41	3.1.2	Bindings and Profiles Document	7
42	3.2	Data Formats.....	8
43	3.2.1	Document Formats	8
44	3.2.2	Signature Formats	8
45	3.3	Core Elements.....	8
46	3.3.1	XML Time-Stamp Token	8
47	3.3.2	Requestor Identity.....	8
48	3.4	Generic Request Requirements	9
49	3.4.1	General	9
50	3.4.2	Input Preparation	9
51	3.4.3	Input Delivery	9
52	3.4.4	Request Identifier	10
53	3.5	Signing Request Requirements.....	10
54	3.5.1	Selective Signing	10
55	3.5.2	Signature Placement	10
56	3.5.3	Claimed Identity.....	10
57	3.5.4	Requested Signing Key	10
58	3.5.5	Intended Audience.....	10
59	3.5.6	Explicit Signed/Unsigned Attributes.....	10
60	3.5.7	Application Profile.....	11
61	3.5.8	Output Options	11
62	3.5.9	Implicit Parameters.....	11
63	3.6	Signing Response Requirements.....	11
64	3.6.1	Signature and Documents.....	11

65	3.7 Verifying Request Requirements.....	12
66	3.7.1 Selective Verification	12
67	3.7.2 Verification Time.....	12
68	3.7.3 Validation Info.....	12
69	3.7.4 Request for Signature Info	12
70	3.7.5 Request for Transformed Data.....	12
71	3.7.6 Request for Signature Verification Steps	12
72	3.7.7 Application Profile.....	12
73	3.7.8 Implicit Parameters	13
74	3.8 Verifying Response Requirements.....	13
75	3.8.1 Failure Information.....	13
76	3.8.2 Signature Info	13
77	3.8.3 Transformed Data.....	13
78	3.8.4 Signature Verification Steps.....	13
79	3.9 Compound Protocol.....	13
80	3.10 Binding Requirements	13
81	3.11 Profile Requirements	14
82	3.11.1 Protocol Profiles	14
83	3.11.2 Signature Profiles	14
84	3.11.3 Binding Profiles	14
85	3.11.4 Application Profiles	14
86	Appendix A. Revision History.....	15
87	Appendix B. Notices	16
88		
89		

89 1 Introduction

90 This document analyses the use cases given in the OASIS DSS Use Case document and
91 identifies the requirements of Digital Signature Services (DSS) protocols. This analysis identifies
92 the particular features of each of the use cases, along with other requirements which have been
93 identified as being necessary to support digital signature services in an open environment.

94 2 Particular Features of Use Cases

95 2.1 Corporate Seal

96 **Submitted by:** Carlisle Adams, Entrust

97 **Message:** <http://lists.oasis-open.org/archives/dss/200301/msg00002.html>

98 **Signature Request:** Server checks requestor as authorized to sign on behalf of the organization

99 **Signature Response:** Signature only identifies organization, not requestor

100 **Verification:** Uses public key belonging to organization

101 2.2 SOAP Signing

102 **Submitted by:** Carlisle Adams, Entrust

103 **Message:** <http://lists.oasis-open.org/archives/dss/200301/msg00002.html>

104 **Signature Request:** Input is XML, requestor may be gateway rather than originator

105 **Signature Response:** XML structure which fits in SOAP structure

106 **Verification:** Uses public key belonging to organization

107 2.3 Identified Requestor

108 **Submitted by:** Carlisle Adams, Entrust

109 **Message:** <http://lists.oasis-open.org/archives/dss/200301/msg00002.html>

110 **Signature Request:** Requestor must be authenticated

111 **Signature Response:** Identifies the requestor

112 **Verification:** Uses public key belonging to organization

113 2.4 Individual Signatures

114 **Submitted by:** Nick Pope

115 **Message:** <http://lists.oasis-open.org/archives/dss/200301/msg00005.html>

116 **Signature Request:** Requestor must be authenticated

117 **Signature Response:** Signature created using key unique to requestor

118 **Verification:** Uses public key belonging to requestor

119 2.5 Long Term Corporate Signatures

120 **Submitted by:** Nick Pope

121 **Message:** <http://lists.oasis-open.org/archives/dss/200301/msg00005.html>

122 **Signature Request:** Input may be existing signed data for re-signing linked to verification
123 **Verification Response:** CRLs / OCSP / Certificate Status Information used to verify signature
124 should be returned with signature for archiving with signature

125 **2.6 Delegated Signature Verification**

126 **Submitted by:** Pieter Kasselmann, Baltimore

127 **Message:** <http://lists.oasis-open.org/archives/dss/200301/msg00010.html>

128 **Verification Request:** Optional specification of signature verification policy

129 **Verification Response:**

- 130 • Returns valid / invalid indicator
- 131 • Return includes information on reason if invalid
- 132 • Return includes signature verification policy applied in validating signature
- 133 • Response signed by server trusted by verification requestor

134 **2.7 Securing the Transform Chain**

135 **Submitted by:** Gregor Karlinger, Austria Federal Ministry for Public Services and Sports

136 **Message:** <http://lists.oasis-open.org/archives/dss/200301/msg00016.html>

137 For use cases where the relying party would like to check the
138 relationship between the the 'transforms process input data'
139 (which is the data he wants to operate on) and the 'transforms
140 process output data' (which is the data the signing party has
141 actually signed) all the information used by the signing party
142 to compute the transforms process must be signed.
143 Most of this information is included in a XMLDSIG signature
144 anyway. However, there are some exceptions, for instance imported
145 stylesheets referred to in an XSLT transform. Those additional
146 information must be signed as well, for instance as part of a dsig:Manifest.

147 **2.8 Non-XML Data Signing**

148 **Submitted by:** Merlin Hughes, Baltimore

149 **Message:** <http://lists.oasis-open.org/archives/dss/200301/msg00036.html>

150 **Signature Response:** S/MIME signatures, signed code (e.g. JAR files), proprietary formats

151 **Verification Request:** S/MIME signatures, signed code (e.g. JAR files), proprietary formats

152 **2.9 ENotarization in presence of notary**

153 **Submitted by:** John Messing

154 **Message:** <http://lists.oasis-open.org/archives/dss/200301/msg00045.html>

155 **Signature Request:**

- 156 • Notary requests signature upon jurat (attestation of witnessing signing and identity and
157 voluntariness of party's signature) and an electronically signed document which is
158 logically associated with the jurat.
- 159 • Server authenticates notary as authorized to sign (e.g. RFC 3039)
- 160 • Server signs jurat and document or its hash on behalf of notary

161 **Signature Response:** Signature created using key of server and logical association of notary
162 with the ultimate requestor
163 **Verification:** Uses public key of server with logical association of notary's authentication as
164 maintained at server

165 **2.10 Court Filings**

166 **Submitted by:** John Messing

167 **Message:** <http://lists.oasis-open.org/archives/dss/200301/msg00047.html>

168 **Signature Request:** Requestor sends document to server (of court, agency or infomediary),
169 which authenticates filer (e.g. RFC 3039)

170 **Signature Response:** Signature created using private key of server PLUS logical association of
171 identity with transaction

172 **Verification:** Uses public key of server PLUS logical, stored association of filer's authentication

173 **2.11 Client-Side Hashing**

174 **Signature Request:** Requestor sends list of dsig:References to be signed

175 **Verification Request:** Requestor sends list of dsig:References to be verified

176 **2.12 Time-stamping / Time-marking**

177 **Signature Response:** Time-mark included in signed data or time-stamp applied to signed data

178 **Verification:** Time-mark or time-stamp included when verifying data

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

3 Requirements

197

3.1 Work Products of the TC

198

3.1.1 Protocols and Core Elements Document

199

- Request/Response Signing Protocol

200

- Request/Response Verifying Protocol

201

- Time-Stamping and Compound Protocols

202

- XML Time-Stamp Token

203

- Requestor Identity Element

204

The TC will develop two request/reponse protocols: one for creating digital signatures, and one for verifying them. These protocols can also be used for symmetric-key MACs, and time-stamps. The TC will also develop an XML Time-Stamp Token, and a Requestor Identity element for use as a signature attribute.

205

206

207

208

The TC will also develop Time-Stamping and Compound Protocols. The Compound Protocol will allow for combinations of the other protocols; however, the details are still cloudy, so it will only be discussed briefly in this document.

209

210

211

In 3.2 we discuss the data formats (documents and signatures) that the protocols take as inputs and outputs.

212

213

In 3.3 we discuss the core elements (the XML Time-Stamp Token and Requestor Identity).

214

In 3.4 we discuss transferring documents from client to server, which is part of both protocols.

215

In 3.5 and 3.6 we discuss the Signing Requests and Responses.

216

In 3.7 and 3.8 we discuss the Verifying Requests and Responses.

217

In 3.9 we discuss Compound Protocols

218

3.1.2 Bindings and Profiles Document

219

Profiles are currently being considered for:

220

- German Signature Law Signature Profile

221

- XAdES Protocol and Signature Profiles

222

- EPM Application Profile

223

- WS-Security Application Profile

224

- S/MIME and Code-Signing Application Profiles

225

- eNotary Application Profile

226

- Corporate Seal Application Profile

227

The TC will develop Bindings for carrying the request/response protocols over different transport and security layers. The TC will also develop Profiles which constrain and extend the Protocols, the Signature Formats they operate on, and their Bindings, to target particular environments. A combination of a Protocol Profile, Signature Profile, and Bindings Profile yields an Application Profile, which will be a fully described service.

228

229

230

231

232

In 3.10 we discuss Bindings.

233

In 3.11 we discuss Profiles.

234

235 **3.2 Data Formats**

236 The Signing Protocol takes as input “documents” and returns as output a “signature”. The
237 Verifying Protocol takes as input both “documents” and a “signature” and returns information on
238 these. We are using the term “document” to refer to any hashable data item, and “signature” to
239 refer to any cryptographic value based on a document hash, such as a public-key signature, a
240 symmetric-key MAC, an RFC 3161 TimeStampToken, or a linked timestamp.

241 **3.2.1 Document Formats**

- 242 • XML
- 243 • Binary

244 A DSS service should be able to apply transforms to, and sign, any sequence of octets. We
245 will consider only XML in particular, and assume everything else can be treated as raw
246 “binary” data without any transforms. Encoding and canonicalization transforms for non-XML
247 data such as HTML, text, multimedia, etc., are beyond the scope of DSS.

248 **3.2.2 Signature Formats**

- 249 • XML-DSIG
- 250 • CMS/PKCS#7 (RFC 2630)
- 251 • XML Time-stamp Token (see 3.3.1)
- 252 • WS-Security Header
- 253 • Extensible to others

254 We will focus on XML-DSIG signatures applied to XML content. As this is the most flexible
255 case and has the most complications around transforms, references, signature placement,
256 etc., the resulting protocol should easily generalize to simpler formats like CMS (which
257 supports the email and code-signing use cases) or OpenPGP.

258

259 **3.3 Core Elements**

260 We will define two XML elements which are useful in conjunction with the request/response
261 protocol. Both of these elements could be used as signature attributes in an XML-DSIG signature.
262 The Time-Stamp Token could also be produced directly, by using the request/response signing
263 protocol to access a DSS server deployed as a TSA.

264 **3.3.1 XML Time-Stamp Token**

265 We will define an XML Timestamp Token, which will be similar to an RFC 3161
266 TimeStampToken, and which can be used for time-stamping XML-DSIG signatures, XML
267 documents, or anything else. We will leave this format extensible so that it can support
268 linking schemes in the future. The representation of time inside this time-stamp token will be
269 a stand-alone XML element that can be reused as a signed attribute to create a “time-
270 marked” signature.

271 **3.3.2 Requestor Identity**

- 272 • Requestor Name (in a type/value format such as a SAML NameIdentifier)
- 273 • (Optionally) Information supporting the name (such as a SAML Assertion, Liberty Alliance
274 Authentication Context, or X.509 Certificate)

275 If the server is not signing with a key specific to the requestor, then the server might want to
276 represent the requestor's name or role, and possibly details of how the requestor
277 authenticated, in a signed attribute. This element is unusual in that it may be used in
278 contexts outside of DSS, such as when a 3rd-party notary signs a document on behalf of a
279 "requestor" who makes his request in person, and authenticates himself with paper
280 documents.
281

282 **3.4 Generic Request Requirements**

283 These requirements apply to both the Signing Request and Verifying Request messages, and
284 deal with how the data to be signed or verified is transferred from client to server.

285 **3.4.1 General**

- 286 • Multiple input documents may be allowed, depending on the signature format
- 287 • Each input may be of a different document format (see 3.2.1)
- 288 • An XML input may include the whole document, or only the part of it that is to be signed
- 289 • ID references need special support

290 An XML-DSIG signature's dsig:References can refer to multiple documents, so the client
291 must be able to transfer multiple documents or portions of documents to the server. Each
292 individual document may be of a different document format (see 3.2.1), and may be prepared
293 and delivered in different ways (see 3.4.2 and 3.4.3). Also, it must be possible for the client
294 to tell the server about schema/DTD information for these documents; otherwise the server
295 cannot know which XML elements have defined ID attributes, and won't be able to create or
296 verify XML-DSIG signatures that refer to document elements via ID references. This
297 information should not be required when referenced XML is in canonical form and well-known
298 identifier attributes are used.

299 **3.4.2 Input Preparation**

- 300 • Unaltered
- 301 • Transformed
- 302 • Hashed

303 The client may send an input document in unaltered form, or may apply some transforms and
304 send the transformed data, or may even apply the transforms, then hash the transformed
305 data, then send the hash. Transferring the document unaltered or transformed allows the
306 server to archive and review what it is signing or verifying. Transferring it hashed is more
307 efficient and preserves client privacy. Applying transforms and hash algorithms on the client
308 side gives the client flexibility to use transforms and hash algorithms the server doesn't know
309 about. When client-side transforms or hashing are applied, the client must tell the server
310 which transforms and which hash algorithms were applied, so the server can represent these
311 in the signature; for example, within a dsig:Reference. The client may also send a list of
312 dsig:References for URIs that contribute to the transform sequence, so that the server may
313 incorporate these into the signature in some fashion.

314 **3.4.3 Input Delivery**

- 315 • Direct
- 316 • Indirect, via URI

317 Each input document may be sent directly, i.e. within the request message itself, or via a URI.

318 **3.4.4 Request Identifier**

- 319 • Arbitrary string which is returned in the response.

320 The client may send an identifier which will be returned in the response, to aid the client in
321 correlating requests and responses.

322

323 **3.5 Signing Request Requirements**

324 These are options the client can use to control how the server produces and returns the
325 signature. In the most general case, the client can include or omit any of these options – if
326 omitted, the server will make his own decisions. Profiles of the request/response protocol may
327 require, or conversely disallow, the use of certain options.

328 If the client requests an option which the server cannot honor, the server will return an error –
329 thus, if the signing request is successful, the client is assured that his entire request was honored.

330 **3.5.1 Selective Signing**

- 331 • Which elements to sign
- 332 • Which transforms to apply
- 333 • Whether the element should be enveloped in the signature

334 The client may specify which particular parts of the documents he sent he wants to sign, how
335 these should be transformed, and whether they should be enveloped inside the signature.

336 **3.5.2 Signature Placement**

- 337 • Point of Insertion
- 338 • Stand-alone Signature

339 If the client wants the signature inserted into one of the input documents, he must specify where.
340 Depending upon the point of insertion, the signature may be considered detached or enveloped
341 with respect to different references.

342 **3.5.3 Claimed Identity**

- 343 • The identity or role asserted by the client.

344 The server may use this to determine signature contents, processing steps, the value of the
345 Requestor Identity element, which key to use, etc..

346 **3.5.4 Requested Signing Key**

- 347 • Which key the document should be signed with.

348 The client may explicitly request a signing key.

349 **3.5.5 Intended Audience**

- 350 • Who the signed document is intended for

351 The client may send a list naming the intended recipients of the signature. The server may
352 use this to modify the implicit parameters (see 3.5.6).

353 **3.5.6 Explicit Signed/Unsigned Attributes**

- 354 • Requestor may provide additional attributes

355 • Server has final say
356 The client may send additional signed and unsigned attributes to include in the signature.
357 The client may send a name/value pair, or may just send a name, and let the server fill in the
358 value (for example, the client may request a time-stamp or time-mark through this
359 mechanism, and let the server fill in the particular time).
360 The server may add other attributes on its own initiative. RFC 3126 on "Long term electronic
361 signatures" is the CMS analogue of XAdES "XML Advanced Electronic Signatures", and both
362 documents contain examples of attributes that might be added.

363 **3.5.7 Application Profile**

364 Which Application Profile to use (see 3.11). If omitted, the server will use its default profile.

365 **3.5.8 Output Options**

366 • Signature by itself
367 • Signature embedded in document (if an enveloped signature was requested)
368 • Untransformed documents
369 • Transformed documents
370 The client may request to receive the signature by itself. If it was an enveloped signature (i.e.
371 *not* a stand-alone signature), the client may also request to receive the document with the
372 signature embedded in it.
373 The client may also request to receive back any of the input documents, either before
374 transforms have been applied (so that the client can send a URI to the server, and then
375 receive back the text of what was signed), or after (so he can view precisely what was signed
376 without having to process the transforms himself).

377 **3.5.9 Implicit Parameters**

378 • What key and validation info is included (certificate chains, CRLs, etc.)
379 • What defaults to use for the other options
380 • What other actions the server should perform (e.g. logging or archiving)
381 • Etc...
382 These parameters, and any others that aren't explicitly dealt with, are implicit in the URI at
383 which the client accesses the server.

384

385 **3.6 Signing Response Requirements**

386 These requirements apply to the message returned by a DSS server after a client requests it to
387 create a signature.

388 **3.6.1 Signature and Documents**

389 • Document with Signature
390 • Stand-alone Signature
391 • Transformed or Untransformed documents
392 Depending on the Output Options (3.5.8) , the server will return the above outputs
393

394 **3.7 Verifying Request Requirements**

395 These requirements apply to the message sent by a client to verify a signature.

396 **3.7.1 Selective Verification**

- 397 • Which signature to verify
- 398 • Whether to perform signature validation, reference validation, or both
- 399 • Which references to verify if performing reference validation
- 400 • Whether to verify dsig:Manifests
- 401 • Which references to verify inside each dsig:Manifest

402 The client may specify parameters identifying the signed information to verify. An XML-DSIG
403 signed document may have multiple signatures, each with multiple references, and these
404 references may include dsig:Manifests. The client may indicate which signature to verify,
405 whether or not references should be verified (and perhaps the client may choose to verify all
406 or some of these himself), and whether or not dsig:Manifests should be verified.

407 **3.7.2 Verification Time**

408 The client may request that the signature be verified at a point in the past, to learn if the
409 signature was valid at a particular date and time.

410 **3.7.3 Validation Info**

411 The client may submit validation info such as certificates, CRLs, and OCSP responses, to
412 help the server verify the signature.

413 **3.7.4 Request for Signature Info**

- 414 • Requestor / Signer identity
- 415 • Signing time
- 416 • Signature policy
- 417 • Signature validity interval
- 418 • Others

419 The client may request the server to process the attributes and keying information associated
420 with the signature and return information about them to the client, to save the client the
421 difficulty of parsing X.509 certificates, SAML Assertions, XAdES attributes, and so on.

422 **3.7.5 Request for Transformed Data**

423 The client may request to receive the transformed data (i.e. the data after the transforms
424 were applied, but before being hashed) to save himself the effort of extricating the data
425 himself and processing the transforms.

426 **3.7.6 Request for Signature Verification Steps**

427 The client may request the server to return a list identifying the steps it (the server) took in
428 verifying the signature (such as checking the signature, validating the path, checking CRLs,
429 etc.). This list can be displayed to the user for informational purposes.

430 **3.7.7 Application Profile**

431 Which Application Profile to use (see 3.11). If omitted, the server will use its default profile.

432 **3.7.8 Implicit Parameters**

- 433 • Which trust settings to use
- 434 • Which resources to consult for trust info (directories, XKMS servers, etc.)
- 435 • How to update the signature (time-stamp, add CRL references, do both, etc.)
- 436 • What granularity of verification steps (3.7.7) to return
- 437 • Etc.

438 These parameters, and any others that aren't explicitly dealt with, are implicit in the URI at
439 which the client accesses the server.

440 **3.8 Verifying Response Requirements**

441 These requirements apply to the message returned by a DSS server after a client requests it to
442 verify a signature.

443 **3.8.1 Failure Information**

- 444 • Reason code
 - 445 • Which references/manifests failed, if appropriate
- 446 The reason codes will differentiate between failures in reference validation, signature
447 validation, an untrusted or revoked signing key, or something unknown/unsupported (a
448 signature format, hash algorithm, transform, verification policy, etc.).

449 **3.8.2 Signature Info**

450 If requested, the server will return signature info (see 3.7.4).

451 **3.8.3 Transformed Data**

452 If requested, the server will return the transformed data (see 3.7.5).

453 **3.8.4 Signature Verification Steps**

454 If requested, the server will return a list of signature verification steps (see 3.7.7).

455

456 **3.9 Compound Protocol**

457 This protocol will support compound operations where a single request/response to a DSS
458 server results in a combination of the sign / verify / timestamp processes.

459 An example of a compound operation is where the client requests the server to verify and
460 then "update" the signature, by time-stamping it, or counter-signing it, or attaching validation
461 info to it, or doing some combination of these (see XAdES or RFC 3126). The intention may
462 be to strengthen the original signature or to make it easier for some other party to verify.

463

464 **3.10 Binding Requirements**

- 465 • Each binding must define a transport (e.g. HTTP, SOAP, BEEP, etc.)
- 466 • Each binding may define confidentiality and integrity means (TLS, WS-Security, etc.)
- 467 • Each binding may define authentication means (passwords, client certificates, etc.)

468

469 **3.11 Profile Requirements**

470 The request/response protocols, signature formats, and bindings, can all be profiled. A profile
471 can limit optionality, instantiate abstractions, add new elements into extensibility points, and add
472 processing rules.

473 A combination of these profiles is an Application Profile that can be used to address a particular
474 use case. DSS server or client implementations will be compliant with particular application
475 profiles of DSS, not with DSS itself.

476 **3.11.1 Protocol Profiles**

- 477 • Supported document and signature formats (see 3.2)
- 478 • Constraints and defaults for optional protocol elements (see 3.5 – 3.8)
- 479 • Addition of new protocol elements to extensibility points

480 A protocol profile will describe which data formats (see 3.2) the protocol supports. In
481 addition, as described in 3.5-3.8, the request/response protocols have a number of optional
482 elements. A protocol profile will either require or disallow some of these elements, and
483 possibly describe their default values (or else leave these as “implicit parameters” of each
484 server). The profile may also extend the protocol with new elements.

485 **3.11.2 Signature Profiles**

486 Signature formats can have different signed and unsigned attributes, including those
487 attributes specific to DSS (like the XML Time-Stamp or Requestor Identity elements), and
488 others. A signature profile describes the allowable signed and unsigned attributes of a
489 signature format, and how the DSS server should handle these attributes in creating and
490 verifying signatures.

491 **3.11.3 Binding Profiles**

492 Bindings may have various options, in particular those related to security. For example, TLS
493 has a variety of ciphersuites, and client certificate authentication can be on or off. A Binding
494 Profile mandates some particular options.

495 **3.11.4 Application Profiles**

496 An Application Profile combines Protocol, Signature, and Binding Profiles so as to produce a
497 fully-described service. Put another way, an Application Profile should be complete enough
498 that independent client and server implementations will interoperate. An Application Profile
499 should take particular care in choosing Binding Profiles so as to achieve sufficient security.

500

501

Appendix A. Revision History

Rev	Date	By Whom	What
Draft-01	2003-3-22	Trevor Perrin	Initial version based on Nick's draft
Draft-02	2003-3-23	Trevor Perrin	Nick's clarifications on time-stamping
Draft-03	2003-4-2	Trevor Perrin	Incorporating feedback from list
Draft-04	2003-4-30	Trevor Perrin	Polish and trying to tie up loose ends
Draft-05	2003-5-18	Trevor Perrin	New text on Work Products (3.10) and Requestor Identity (3.2.1)
Draft-06	2003-5-18	Trevor Perrin	Updated 3.9 to new definition of "Protocol Bindings"
Draft-07	2003-6-25	Trevor Perrin	Miscellaneous
Draft-08	2003-7-12	Trevor Perrin	Refactored much of the document
Draft-09	2003-7-22	Trevor Perrin	Frederick's feedback
Draft-10	2003-7-29	Trevor Perrin	After Face-to-Face meeting
Draft-11	2003-8-14	Trevor Perrin	Potential final version?
Draft-12	2003-8-16	Trevor Perrin	Removed sentence on time-stamping

Appendix B. Notices

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

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

516 Copyright © The Organization for the Advancement of Structured Information Standards [OASIS]
517 2002 and 2003. All Rights Reserved.

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

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

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