



SAML V2.0 Condition for Delegation Restriction Version 1.0

Committee Draft 02

22 September 2009

Specification URIs:

This Version:

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-delegation-cd-02.html>
<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-delegation-cd-02.odt> (Authoritative)
<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-delegation-cd-02.pdf>

Previous Version:

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-delegation-cd-01.html>
<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-delegation-cd-01.odt> (Authoritative)
<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-delegation-cd-01.pdf>

Latest Version:

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-delegation.html>
<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-delegation.odt> (Authoritative)
<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-delegation.pdf>

Technical Committee:

OASIS Security Services TC

Chair(s):

Hal Lockhart, Oracle Corporation
Thomas Hardjono, M.I.T.

Editors:

Scott Cantor, Internet2

Declared XML Namespace(s):

urn:oasis:names:tc:SAML:2.0:conditions:delegation

Abstract:

This document defines a `<saml:Condition>` type for expressing a chain of intermediaries acting on behalf of the subject of an assertion, requiring relying parties to distinguish between direct and indirect access.

32 **Status**

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

36 TC members should send comments on this specification to the TC's email list. Others
37 should send comments to the TC by using the "Send A Comment" button on the TC's
38 web page at <http://www.oasis-open.org/committees/security>.

39 For information on whether any patents have been disclosed that may be essential to
40 implementing this specification, and any offers of patent licensing terms, please refer to the IPR
41 section of the TC web page (<http://www.oasis-open.org/committees/security/ipr.php>).

42 The non-normative errata page for this specification is located at [http://www.oasis-](http://www.oasis-open.org/committees/security)
43 [open.org/committees/security](http://www.oasis-open.org/committees/security).

44 Notices

45 Copyright © OASIS Open 2009. All Rights Reserved.

46 All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual
47 Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

48 This document and translations of it may be copied and furnished to others, and derivative works that
49 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published,
50 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice
51 and this section are included on all such copies and derivative works. However, this document itself may
52 not be modified in any way, including by removing the copyright notice or references to OASIS, except as
53 needed for the purpose of developing any document or deliverable produced by an OASIS Technical
54 Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be
55 followed) or as required to translate it into languages other than English.

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

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

63 OASIS requests that any OASIS Party or any other party that believes it has patent claims that would
64 necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to
65 notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such
66 patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced
67 this specification.

68 OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any
69 patent claims that would necessarily be infringed by implementations of this specification by a patent
70 holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR
71 Mode of the OASIS Technical Committee that produced this specification. OASIS may include such
72 claims on its website, but disclaims any obligation to do so.

73 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
74 might be claimed to pertain to the implementation or use of the technology described in this document or
75 the extent to which any license under such rights might or might not be available; neither does it represent
76 that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to
77 rights in any document or deliverable produced by an OASIS Technical Committee can be found on the
78 OASIS website. Copies of claims of rights made available for publication and any assurances of licenses
79 to be made available, or the result of an attempt made to obtain a general license or permission for the
80 use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS
81 Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any
82 information or list of intellectual property rights will at any time be complete, or that any claims in such list
83 are, in fact, Essential Claims.

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

88 **Table of Contents**

89 1 Introduction.....5
90 1.1 Notation.....5
91 1.2 Normative References.....6
92 1.3 Non-Normative References.....6
93 2 SAML V2.0 Condition for Delegation Restriction.....7
94 2.1 Required Information.....7
95 2.2 Overview.....7
96 2.2.1 Terminology and Motivation.....7
97 2.3 Element <Delegate>.....9
98 2.4 Complex Type DelegationRestrictionType.....10
99 2.5 Use of Identifiers Within <saml:SubjectConfirmation>.....10
100 2.6 Security Considerations.....11
101 3 Conformance.....12
102 3.1 SAML V2.0 Condition for Delegation Restriction.....12
103 Appendix A. Acknowledgements.....13
104 Appendix B. Revision History.....14
105

1 Introduction

Some advanced SAML use cases involve a single logical transaction that spans one or more intermediate clients or servers. A common example includes a SAML-enabled web site acting on behalf of a logged-in user while accessing additional SAML-enabled web services. Generalizing this example, a number of intermediaries might be transited before the final point of access. If a SAML assertion is used as a security token to authenticate and authorize such access, it is important that the identity and order of intermediaries, if any, be expressed within the token in some fashion.

Existing mechanisms designed for this purpose, such as the `<saml:SubjectConfirmation>` element definition in the SAML V2.0 core specification [SAML2Core], or the extended syntax found in the Liberty ID-WSF Security Mechanisms specification [LibSecMech20], suffer from the drawback that they have advisory semantics for a relying party and are likely to be ignored by delegation-unaware SAML processing. While backward compatibility can be an advantage, ignoring security-relevant details that might impact upon a relying party's policy is unacceptable in some scenarios.

This specification provides for the expression of delegation information with normative SAML processing semantics through the use of a `<saml:Condition>` extension type.

1.1 Notation

This specification uses normative text.

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in [RFC2119]:

...they MUST only be used where it is actually required for interoperation or to limit behavior which has potential for causing harm (e.g., limiting retransmissions)...

These keywords are thus capitalized when used to unambiguously specify requirements over protocol and application 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.

Listings of XML schemas appear like this.

Example code listings appear like this.

Conventional XML namespace prefixes are used throughout the listings in this specification to stand for their respective namespaces as follows, whether or not a namespace declaration is present in the example:

Prefix	XML Namespace	Comments
saml:	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace defined in the SAML V2.0 core specification [SAML2Core].
del:	urn:oasis:names:tc:SAML:2.0:conditions:delegation	This is the namespace defined by this specification.
xsd:	http://www.w3.org/2001/XMLSchema	This namespace is defined in the W3C XML Schema specification [Schema1]. In schema listings, this is the default namespace and no prefix is shown.
xsi:	http://www.w3.org/2001/XMLSchema-instance	This is the XML Schema namespace for schema-related markup that appears in XML instances [Schema1].

137 This specification uses the following typographical conventions in text: <SAML*E*lement>,
138 <ns:ForeignElement>, Attribute, **Datatype**, OtherCode.

139 1.2 Normative References

- 140 **[Delegation-XSD]** OASIS Committee Draft 02, *SAML V2.0 Condition for Delegation Restriction*
141 *Schema*, September 2009. [http://docs.oasis-open.org/security/saml/Post2.0/sstc-](http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-delegation.xsd)
142 [saml-delegation.xsd](http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-delegation.xsd)
- 143 **[RFC2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF
144 RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>
- 145 **[SAML2Core]** OASIS Standard, *Assertions and Protocols for the OASIS Security Assertion*
146 *Markup Language (SAML) V2.0*, March 2005. [http://docs.oasis-](http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf)
147 [open.org/security/saml/v2.0/saml-core-2.0-os.pdf](http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf)
- 148 **[Schema1]** H. S. Thompson et al. *XML Schema Part 1: Structures*. World Wide Web
149 Consortium Recommendation, May 2001. [http://www.w3.org/TR/2001/REC-](http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/)
150 [xmlschema-1-20010502/](http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/)
- 151 **[Schema2]** Paul V. Biron, Ashok Malhotra. *XML Schema Part 2: Datatypes*. World Wide Web
152 Consortium Recommendation, May 2001. [http://www.w3.org/TR/2001/REC-](http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/)
153 [xmlschema-2-20010502/](http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/)

154 1.3 Non-Normative References

- 155 **[LibSecMech20]** F.Hirsch. *Liberty ID-WSF Security Mechanisms Core*. November 2006.
156 <http://www.projectliberty.org/specs>

2 SAML V2.0 Condition for Delegation Restriction

2.1 Required Information

Identification: urn:oasis:names:tc:SAML:2.0:conditions:delegation

Contact information: security-services-comment@lists.oasis-open.org

Description: Given below.

Updates: None.

2.2 Overview

The SAML V2.0 core specification [SAML2Core] defines the **saml:ConditionAbstractType** complex type as a basis for extensions with mandatory processing semantics for relying parties. This specification defines such an extension as a supplement for the presence of an identifier within the `<saml:SubjectConfirmation>` element.

Rather than an advisory mechanism for identifying a single delegate, the extension provides for a normative mechanism that identifies an ordered sequence of delegates, along with optional detail about the acts of delegation.

2.2.1 Terminology and Motivation

Delegation can be complex and is frequently conflated, combined, or confused with a number of related approaches. Without attempting to address all the myriad ways of describing such interactions, for the purposes of this profile the following is an attempt to capture some of the alternatives encountered and how the notion of delegation is meant for the purposes of this profile. In most of the cases presented, the flows involved are simplified for illustration. These are not meant as normative scenarios.

Proxying

As described by section 3.4.1.5 of the SAML V2.0 core specification [SAML2Core], proxying occurs when an intermediate identity provider issues an assertion to a relying party on the basis of an assertion issued to it. Proxying is a gateway-like function in which the subject of the assertion is presumed to directly interact with each party directly.

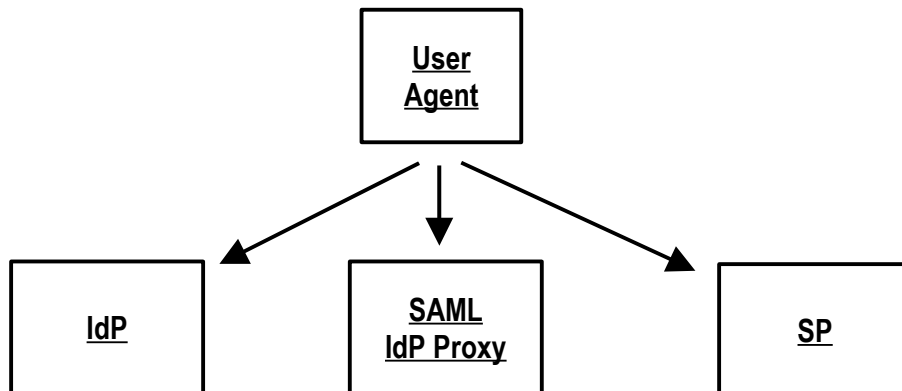
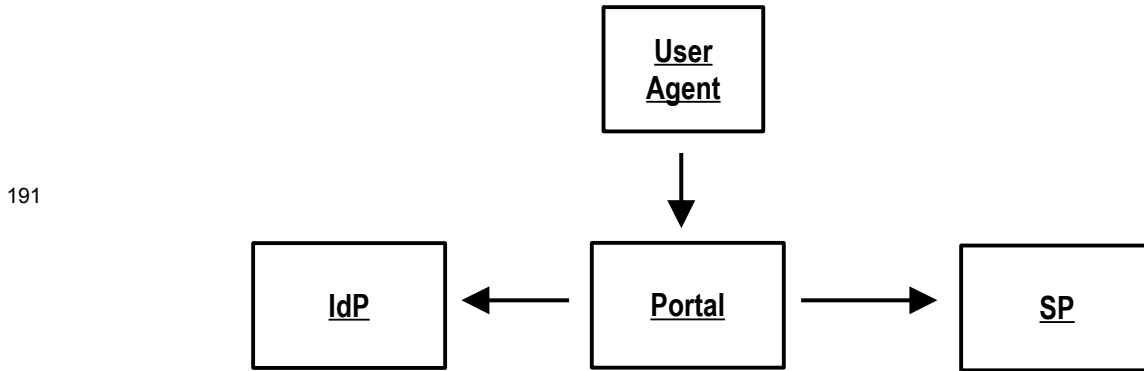


Figure 1: Proxying

In Figure 1: Proxying, the sequence proceeds from left to right, with the user agent authenticating to the left-most system to obtain an assertion for the proxy, which issues an assertion for the user agent to deliver to the service.

186 Impersonation

187 An impersonation model is one in which an entity acting on behalf of an assertion subject is able
188 to obtain and use an assertion indistinguishable from an assertion that would be issued directly to
189 the subject. A typical example of such a scenario might include a portal that holds credentials for
190 users and is able to authenticate directly to obtain assertions about them.

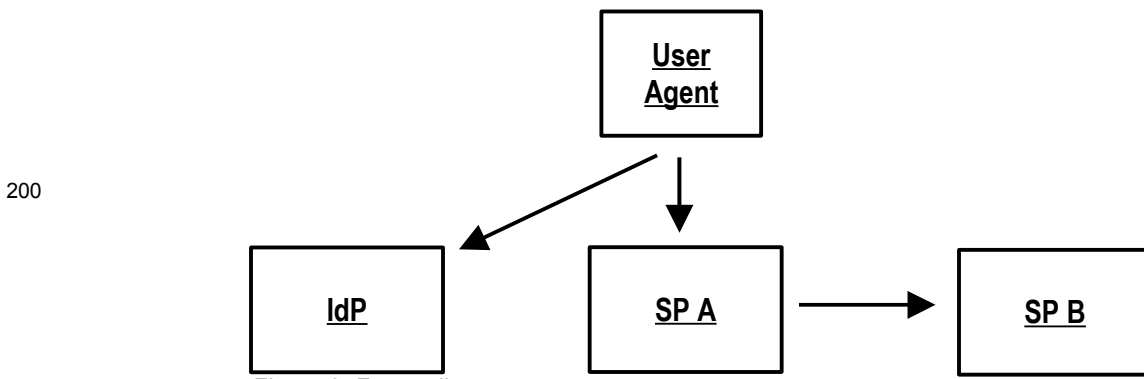


191
192
193
194
Figure 2: Impersonation

192 In Figure 2: Impersonation, the user agent supplies its credentials (e.g., a password) to the portal,
193 which relays them to the identity provider on the left, obtaining an assertion that it supplies to the
194 service. As far as the service knows, the portal is a direct user agent.

195 Forwarding

196 Forwarding is a form of impersonation in which an assertion is directly reused by an intermediary
197 to impersonate a subject from whom an assertion was obtained. It is distinguished from
198 delegation in that assertions are passed along unchanged, and because (as an impersonation
199 model) the assertion is not modified in a fashion that would identify the intermediary.



200
201
202
203
204
205
Figure 3: Forwarding

202 In Figure 3: Forwarding, the user agent authenticates directly to the identity provider, obtaining an
203 assertion that it delivers to the service in the middle. The service directly forwards the assertion to
204 the second service on the right, acting as the subject. As far as the second service knows, the
205 first service is a direct user agent (as with impersonation).

206 Delegation

207 Finally, delegation moves beyond the forwarding scenario by adding information to the assertion
208 that explicitly identifies all the parties through which a transaction flows. While it is ultimately a
209 matter for the identity provider as to how and on what basis this information is collected, it is often
210 assured by routing requests back through the identity provider at each hop to cryptographically
211 guarantee that each party has been authenticated and appropriate policy enforced. This fine-
212 grained and real-time enforcement capability is a key advantage over pure forwarding or
213 impersonation.

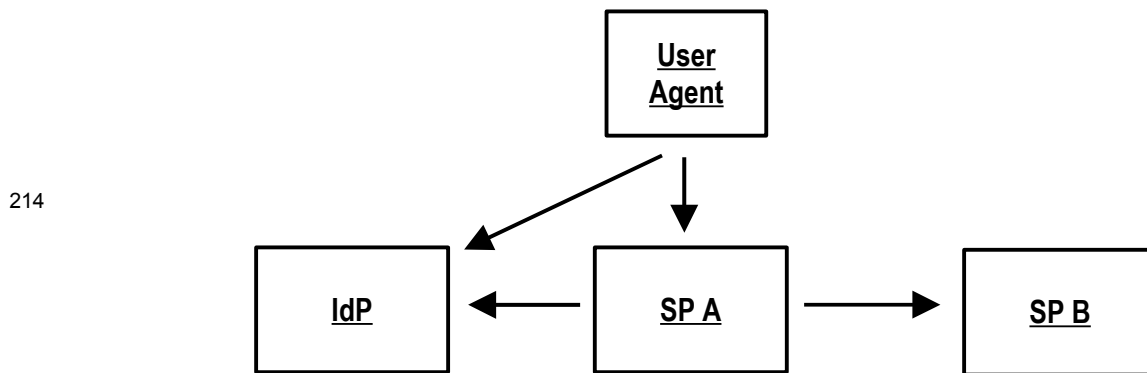


Figure 4: Delegation

215 In Figure 4: Delegation, the forwarding model is extended by adding a request back to the identity
216 provider by the intermediary service in the middle. This allows a new assertion to be issued to it
217 that, while it may identify the same subject as the original, also identifies the service as a delegate
218 of the subject. This identification can be performed advisedly, in a manner defined by
219 [SAML2Core], or with a normative semantic as defined by this profile.

220 So, in summary, this profile is intended to address scenarios in which assertions are materially altered to
221 reflect the path of a transaction through one or more intermediaries that act on behalf of the subject of the
222 assertion. These intermediaries are termed *delegates*, and an assertion carrying the condition type
223 defined in this profile is termed a *delegate assertion*. The act of producing such an assertion is then
224 termed *delegation* and we can say that the credentials from which the assertion is derived have been
225 *delegated*. Such credentials are therefore *delegatable*. In the context of SAML, an assertion might be used
226 as a delegatable credential, so it's possible in multiple-tier scenarios for a delegate assertion to itself be
227 delegatable.

228 There are no normative requirements associated with the use of these terms, and they do not materially
229 affect the semantics of the profile, but using terms consistently across implementations and scenarios is
230 likely to aid understanding and deployment.

231 2.3 Element <Delegate>

232 The <Delegate> element is a container for a single intermediary/delegate represented by the assertion.
233 It contains the following elements and attributes:

234 DelegationInstant [Optional]

235 A timestamp indicating the approximate time at which the act of delegation occurred, if known.

236 ConfirmationMethod [Optional]

237 Identifies the subject confirmation method used, if the delegate presented a SAML assertion to
238 authenticate itself to the issuing authority.

239 <saml:BaseID>, <saml:NameID>, <saml:EncryptedID> [Required]

240 Identifies the delegate.

241 The delegate is identified by a required child element in the usual SAML fashion. The optional attributes, if
242 present, supply additional information about the act of delegation.

243 The following schema fragment defines the <Delegate> element and its **DelegateType** complex type:

```
244 <element name="Delegate" type="del:DelegateType"/>
245 <complexType name="DelegateType">
246   <choice>
247     <element ref="saml:BaseID"/>
248     <element ref="saml:NameID"/>
249     <element ref="saml:EncryptedID"/>
250   </choice>
251   <attribute name="DelegationInstant" type="dateTime" use="optional"/>
252   <attribute name="ConfirmationMethod" type="anyURI" use="optional"/>
253 </complexType>
```

254 2.4 Complex Type DelegationRestrictionType

255 The **DelegationRestrictionType** complex type defines a subtype of **saml:ConditionType** representing
256 one or more acts of delegation that are represented by the containing assertion. It contains the following
257 elements:

258 <Delegate> [One or more]

259 An element identifying a delegate of the subject of the containing assertion. The delegates **MUST** be
260 ordered from least to most recent; thus the earliest element is the farthest removed from the
261 immediate use of the assertion.

262 A relying party **MUST** evaluate the list of delegates, and **SHOULD NOT** accept the assertion unless it
263 wishes to permit each delegate to act on behalf of the subject of the containing assertion.

264 A SAML authority **MUST NOT** include more than one <saml:Condition> element of this type within a
265 <saml:Conditions> element of an assertion.

266 For the purposes of determining the validity of the <saml:Conditions> element, this condition type is
267 always considered to be valid. That is, this condition type does not affect assertion validity, but is a
268 condition on use.

269 The following schema fragment defines the **DelegationRestrictionType** complex type:

```
270 <complexType name="DelegationRestrictionType">
271   <complexContent>
272     <extension base="saml:ConditionAbstractType">
273       <sequence>
274         <element ref="del:Delegate" maxOccurs="unbounded"/>
275       </sequence>
276     </extension>
277   </complexContent>
278 </complexType>
```

279 2.5 Use of Identifiers Within <saml:SubjectConfirmation>

280 For consistency with the existing SAML-defined syntax, it is **RECOMMENDED** that the identifier of the
281 most recent delegate (within the last element in the condition, per section 2.4) be duplicated within the
282 relevant <saml:SubjectConfirmation> elements in the containing assertion.

283 **2.6 Security Considerations**

284 The content of this condition type is directly impacted by the security semantics of the flow of activity that
285 leads to the issuance of the containing assertion. This specification does not define the exchanges that
286 must take place, and relies on composition with other profiles that logically represent acts of delegation
287 that require representation in an assertion.

288 Relying parties are not required to apply any particular policies with regard to the information represented
289 by this condition type. Rather, it is expected that such information will naturally be significant in the
290 enforcement of existing policies, and that the presence of delegation is significant enough to warrant the
291 disruption of existing services designed to consume SAML assertions until those policies reflect a
292 willingness to accept more indirect forms of access.

293 **3 Conformance**

294 **3.1 SAML V2.0 Condition for Delegation Restriction**

295 An assertion issuer conforms to this specification if it can generate assertions containing a
296 `<saml:Condition>` of type **DelegationRestrictionType**, per section 2.

297 A relying party conforms to this specification if it can successfully process assertions containing a
298 `<saml:Condition>` of type **DelegationRestrictionType**, per section 2.

Appendix A. Acknowledgements

300 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
301 Committee, whose voting members at the time of publication were:

- 302 • Rob Philpott, EMC Corporation
- 303 • Richard Franck, IBM
- 304 • John Bradley, Individual
- 305 • Scott Cantor, Internet2
- 306 • Nate Klingenstein, Internet2
- 307 • Bob Morgan, Internet2
- 308 • Thomas Hardjono, M.I.T.
- 309 • Tom Scavo, National Center for Supercomputing Applications (NCSA)
- 310 • Frederick Hirsch, Nokia Corporation
- 311 • Paul Madsen, NTT Corporation
- 312 • Ari Kermaier, Oracle Corporation
- 313 • Hal Lockhart, Oracle Corporation
- 314 • Anil Saldhana, Red Hat
- 315 • Kent Spaulding, Skyworth TTG Holdings Limited
- 316 • Duane DeCouteau, Veterans Health Administration
- 317 • David Staggs, Veterans Health Administration

318 **Appendix B. Revision History**

- 319 ● Draft 01
- 320 ● Committee Draft 01, CD edits
- 321 ● Draft 02, additional explanatory text following public review
- 322 ● Committee Draft 02, CD edits