



# Metadata for SAML 2.0 Web Browser SSO Profiles

Working Draft 00, 15 September 2003

**Document identifier:**

sstc-saml-metadata-2.0-draft-00

**Location:**

<http://www.oasis-open.org/apps/org/workgroup/security/download.php>

**Editor:**

Jahan Moreh, Sigaba <jmoreh@sigaba.com>

**Contributors:**

Prateek Mishra, Netegrity  
Jeff Hodges, Sun Microsystems  
Charles Knouse, Oblix  
Rob Philpott, RSA  
Frederick Hirsch, Nokia  
Tim Moses, Entrust

**Abstract:**

The SAML Web Browser SSO Profiles require agreements between source and destination sites about information such as URLs, source and destination IDs, certificates and keys, and so forth. Metadata definitions are useful for describing this information in a standardized way. This document defines metadata that describe the elements and attributes required to use the SAML Web Browser SSO Profiles. Since the Liberty Alliance Web SSO Profiles are directly based on the SAML Web SSO Profiles, the metadata defined in this document borrows extensively from the metadata definitions in the draft Liberty Alliance 1.2 specifications.

**Status:**

Interim draft. Send comments to the editor.

Committee members should send comments on this specification to the [security-services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list. Others should subscribe to and send comments to the [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org) list. To subscribe, send an email message to [security-services-comment-request@lists.oasis-open.org](mailto:security-services-comment-request@lists.oasis-open.org) with the word "subscribe" as the body of the message.

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 Security Services TC web page (<http://www.oasis-open.org/committees/security/>).

---

38 **Table of Contents**

39 1 Introduction ..... 3  
40 1.1 Notation ..... 3  
41 1.2 Trust Model ..... 3  
42 2 Metadata for SAML Web Browser SSO Profiles ..... 4  
43 2.1 Schema declaration ..... 4  
44 2.1.1 Namespaces in metadata ..... 4  
45 2.1.2 Data type `entityIDType`..... 5  
46 2.1.3 Common attributes..... 5  
47 2.1.4 Common elements..... 5  
48 2.1.5 Base type `providerDescriptorType` ..... 6  
49 2.1.6 Element `IDPDescriptor`..... 8  
50 2.1.7 Element `SPDescriptor`..... 9  
51 2.2 Entity Descriptors ..... 10  
52 2.2.1 Element `EntityDescriptor` ..... 10  
53 3 Examples ..... 11  
54 3.1 Identity Provider ..... 11  
55 3.2 Service Provider ..... 11  
56 4 Validating Metadata Over An Untrusted Channel..... 14  
57 4.1 Introduction..... 14  
58 4.2 Procedure..... 14  
59 4.3 Validation string calculation..... 14  
60 5 References..... 16  
61 Appendix A. Revision History ..... 17  
62 Appendix B. Issues list and resolution..... 18  
63 Appendix C. TO Dos..... 20  
64 Appendix D. Notices ..... 21

---

65

---

## 66 1 Introduction

67 | The SAML Web Browser SSO Profiles [\[SAMLBind\]](#) require agreement between a source and  
68 | destination site about supported protocols, URL of assertion producers and consumers, source  
69 | and destination IDs, certificates and keys, and so forth. Sources and destinations of SAML Web  
70 | Browser SSO Profiles can exchange this information via a set of metadata as defined in this  
71 | document.

72

73 | The Liberty Alliance 1.2 draft specifications include metadata description and discovery protocols  
74 | [\[libMD\]](#). This document specifies elements and attributes from the Liberty 1.2 draft specification  
75 | that are used to describe the required metadata for entities using the SAML Web Browser SSO  
76 | Profiles.

### 77 1.1 Notation

78 | The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”,  
79 | “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this specification are to be  
80 | interpreted as described in IETF RFC 2119 [\[RFC2119\]](#).

81 | `Listings of productions or other normative code appear like this.`

82

83 | `Example code listings appear like this.`

84

Note: Non-normative notes and explanations appear like this.

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

88 | The prefix `xsd:` stands for the W3C XML Schema namespace

89 | The prefix `saml:` stands for the SAML assertion namespace [\[SAMLCore\]](#).

90 | The prefix `samlp:` stands for the SAML request-response protocol namespace [\[SAMLCore\]](#).

91 | The prefix `ds:` stands for the W3C XML Signature namespace,

92 | <http://www.w3.org/2000/09/xmlsig#> [\[XMLSig\]](#).

### 93 1.2 Trust Model

94 | This document specifies a single trust model between assertion producers and assertion  
95 | consumers. This trust model is based on exchanging public keys in the metadata. These public  
96 | keys, which may be contained in a standard X.509 certificate, are used by assertion consumers  
97 | to verify assertions requests, or responses that may be signed by assertion producers. Additional  
98 | trust models and corresponding credentials are beyond the scope of this specification.

99

## 2 Metadata for SAML Web Browser SSO Profiles

100 The Liberty metadata discovery protocol provides for real-time exchanges of metadata between  
101 *providers*. SAML 2.0 does not use Liberty's publishing protocols for real-time exchange of  
102 metadata. Rather, SAML source and destination sites must *a priori* have obtained metadata  
103 regarding each other. That is, the protocol for exchanging metadata is outside the scope of this  
104 specification.

105

106 The SAML 2.0 Metadata specification adopts Liberty's terminology for *providers* as follows:

- 107     ▪ Identity Provider (IDP) means the *source* of an authentication context. The source  
108       performs initial user authentication.
- 109     ▪ Service Provider (SP) means the *destination* that provides services and requires an  
110       authentication context.

111

112 The metadata schema described in this specification allows for a single method of representation.  
113 This representation consists of a single document expressing all of the metadata for a single  
114 provider identified by one or more `providerIDs`. Note that both the IDP and the SP are usually  
115 *producers* and *consumers* of metadata.

116

117 The metadata schema in this specification **does not allow** for single document describing  
118 multiple, distinct providers identified by multiple `providerIDs`, or multiple documents which  
119 individually describe portions of a single provider's metadata.

120

121 Multiple `providerIDs` may be used to identify a single provider. This allows multiple unique  
122 identifiers for the provider. This is supported in the metadata schema by using multiple  
123 `SPDescriptors` (or multiple `IDPDescriptors`) in a metadata description, each associated  
124 with a single `providerID`.

### 2.1 Schema declaration

126 The primary container for a published metadata document is `EntityDescriptor` (see Section  
127 2.2.1 for more details). The immediate child node of `EntityDescriptor` expects one or more  
128 of:

129     `IDPDescriptor` (see Section 2.1.6 for more details)

130     `SPDescriptor` (see Section 2.1.7 for more details)

#### 2.1.1 Namespaces in metadata

132 The SAML metadata structures are modeled after the metadata schema specified by the Liberty  
133 Alliance Project (see [\[libMD\]](#) for additional details). The namespaces referenced in [\[libMD\]](#) that  
134 are used in SAML metadata structures are:

135     `ds:` is described by the W3C XML signature schema [\[XMLSig\]](#)

136     `saml:` is described by SAML assertion namespace [\[SAMLCore\]](#)

137

138 The namespaces referenced in [\[libMD\]](#) that are not used in SAML metadata structures are:

139     `isf:` the Liberty Services Framework schema

140     `pip:` the Liberty Personal Information Profile schema

141

142 The following schema fragment illustrates the use of namespaces in SAML metadata documents:

```
143 <xsd:schema targetNamespace="urn:oasis:names:tc:SAML:2.0:metadata"
144 xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
145 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
146 xmlns:xs="http://www.w3.org/2001/XMLSchema"
147 xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
148
```

## 149 2.1.2 Data type `entityIDType`

150 The data type `entityIDType` is adopted without modification from [\[libMD\]](#). Basically,  
151 `entityIDType` restricts the XML schema datatype `anyURI` to a length of 1024 bytes. Within this  
152 specification `entityIDType` is used to describe a unique ID attribute for providers. The schema  
153 fragment for `entityIDType` is:

```
154 <xsd:simpleType name="entityIDType">
155   <xsd:restriction base="xsd:anyURI">
156     <xsd:maxLength id="maxlengid" value="1024" />
157     <xsd:pattern value="" />
158   </xsd:restriction>
159 </xsd:simpleType>
```

## 160 2.1.3 Common attributes

161 SAML metadata specification adopts two common attributes from [\[libMD\]](#):

162

163 `providerID` of type `entityIDType` indicates the `providerID` of the provider, described by the  
164 children of the node.

165

166 `validUntil` of type `dateTime` indicates expiration date and time of the node and its  
167 descendants. If `dateTime` expressions evaluate to nonequivalent values, parsers MUST adhere  
168 to the most restrictive value (i.e., the earliest `dateTime`).

169

170 The schema fragment for `providerID` and `validUntil` is:

```
171 <xsd:attribute name="providerID" type="entityIDType" />
172 <xsd:attribute name="validUntil" type="xsd:dateTime" />
173
```

## 174 2.1.4 Common elements

175 SAML 2.0 metadata specification adopts common elements from [\[libMD\]](#) as described in the  
176 following subsections.

### 177 2.1.4.1 Element `Extension`

178 The `Extension` element allows for arbitrary content extensions from other namespaces.  
179 Providers can use this element to exchange any metadata that is not defined by this specification.  
180 The schema fragment for `Extension` is:

181

```
182 <xsd:element name="Extension" type="extensionType" />
```

```
183 <xsd:complexType name="extensionType">
184 <xsd:sequence >
185 <xsd:any namespace="##other" processContents="lax" maxOccurs="unbounded"
186 />
187 </xsd:sequence>
188 </xsd:complexType>
```

#### 189 2.1.4.2 Element Organization

190 The Organization element provides some basic information about the provider. This optional  
191 element is informative in nature and does not directly map to any SAML elements or attributes.  
192 The Organization element consists of the following:

193

194 Element OrganizationName of type string specifies the organizational name of the provider.  
195 Exactly one OrganizationName MUST be specified.

196

197 Element OrganizationDisplayName of type string specifies the organizational name of the  
198 provider suitable for display. Exactly one OrganizationDisplayName MUST be specified.

199

200 Element OrganizationURL of type anyURI specifies a URL for the organization and may be  
201 used to direct a principal to the provider for additional information. Exactly one  
202 OrganizationURL MUST be specified.

203

204 Element Extension of type extensionType allows for inclusion of additional arbitrary  
205 information about the organization. This is an optional element.

206

207 The schema fragment for Organization is:

```
208 <xsd:element name="Organization" type="organizationType" />
209 <xsd:complexType name="organizationType">
210 <xsd:sequence>
211 <xsd:element name="OrganizationName" type="xsd:string" />
212 <xsd:element name="OrganizationDisplayName" type="xsd:string" />
213 <xsd:element name="OrganizationURL" type="xsd:anyURI" />
214 <xsd:element ref="Extension" minOccurs="0" />
215 </xsd:sequence>
216 </xsd:complexType>
```

#### 217 2.1.5 Base type providerDescriptorType

218 The providerDescriptorType generically describes metadata for SAML identity and service  
219 providers. The providerDescriptorType is extended by IDPDescriptorType and  
220 SPDescriptorType, which further describe specific metadata about identity and service  
221 providers respectively (See Sections 2.1.6 and 2.1.7 for more details).

222

223 The type providerDescriptorType consists of the following attributes and elements:

224 providerID [required]

225 validUntil [required]

226 protocolSupportEnumeration [required]

227 id [optional]

228 ds:keyInfo [optional]  
229 Organization [optional]  
230 Extension [optional]  
231 ds:Signature [optional]

232

233 The schema fragment for providerDescriptorType is:

```
234 <xsd:complexType name="providerDescriptorType">  
235   <xsd:sequence>  
236     <xsd:element ref="ds:keyInfo" minOccurs="0" />  
237     <xsd:element ref="Organization" minOccurs="0" />  
238     <xsd:element ref="Extension" minOccurs="0" />  
239     <xsd:element ref="ds:Signature" minOccurs="0" />  
240   </xsd:sequence>  
241   <xsd:attribute ref="providerID" use="required" />  
242   <xsd:attribute ref="validUntil" use="required" />  
243   <xsd:attribute name="protocolSupportEnumeration" type="xsd:NMTOKENS"  
244   use="required" />  
245   <xsd:attribute name="id" type="xsd:ID" use="optional" />  
246 </xsd:complexType>
```

### 247 2.1.5.1 Attribute providerID

248 The attribute providerID specifies a unique ID for the provider. The providerID MUST be  
249 unambiguous to the entities that process the metadata (since the providerID is a URI, the  
250 designated value should be unambiguous regardless of context).

251

252 The following additional rule applies to Identity Providers whose saml:issuer attribute is a URI.  
253 The providerID attribute MUST be the same as the saml:issuer attribute specified in  
254 Section 2.3 of [\[SAMLCore\]](#). Note that the preceding rule is consistent with the specification in  
255 Section 2.3 of [\[SAMLCore\]](#) wherein the normative specification for saml:issuer is: *The issuer*  
256 *name*

257 SHOULD be unambiguous to the intended relying parties. SAML authorities may use an identifier  
258 such as a URI reference that is designed to be unambiguous regardless of context.

### 259 2.1.5.2 Attribute protocolSupportEnumeration

260 The attribute protocolSupportEnumeration describes the protocol release type of the  
261 provider described by providerID. This attribute is of the type NMTOKEN, which allows for the  
262 enumeration of a set of SAML protocol releases which the provider MUST support. The data type  
263 of the tokens MUST be URNs and the value must be one of the following:

264 urn:oasis:names:tc:SAML:1.0:protocol  
265 urn:oasis:names:tc:SAML:1.1:protocol  
266 urn:oasis:names:tc:SAML:2.0:protocol

### 267 2.1.5.3 Attribute validUntil

268 The attribute validUntil indicates the date and time beyond which the metadata can not be  
269 guaranteed as being valid. Entities that process a provider's metadata MUST reject the metadata  
270 as stale if the value of this attribute evaluates to a date and time before the current date and time.

#### 271 **2.1.5.4 Attribute `id`**

272 The attribute `id` specifies the fragment identifier for the instance node. This attribute is optional,  
273 unless the node is being signed. For signed nodes an `id` attribute MUST be specified.

#### 274 **2.1.5.5 Element `ds:keyInfo`**

275 | The element `ds:keyInfo` contains the provider's public key in a format allowed by [XMLSig](#).  
276 The provider's public key is used for verifying signed assertions, requests, and responses.

#### 277 **2.1.5.6 Element `Organization`**

278 This element is used to provide information about the provider's organization. See Section 2.1.4.2  
279 for details.

#### 280 **2.1.5.7 Element `Extension`**

281 This element may be used to specify additional metadata about the provider. An entity that does  
282 not understand the value of this element SHOULD reject the metadata as unacceptable.

#### 283 **2.1.5.8 Element `ds:Signature`**

284 A provider may digitally sign its metadata. The element `ds:Signature` specifies a signature that  
285 | is conformant with the specification in [XMLSig](#). Note that the public key that verifies the  
286 signature of the metadata document must be exchanged out of band. That is, the public key in  
287 the metadata document, as described in Section 2.1.5.5 SHOULD only be used for verifying  
288 assertions, requests, and responses.

### 289 **2.1.6 Element `IDPDescriptor`**

290 The element `IDPDescriptor` is of type `IDPDescriptorType`, which extends  
291 `providerDescriptorType` with the following attributes and elements:

292 `SoapEndPoint` [required]

293 `SingleSignonServiceURL` [required]

294 `SingleSignonProtocolProfile` [required]

295 `IssuerID` [optional]

296

297 The schema fragment for `IDPDescriptor` is:

```
298 <xsd:complexType name="IDPDescriptorType">  
299   <xsd:complexContent>  
300     <xsd:extension base="providerDescriptorType">  
301       <xsd:sequence>  
302         <xsd:element name="SoapEndpoint" type="xsd:anyURI" />  
303         <xsd:element name="SingleSignonServiceURL" type="xsd:anyURI" />  
304         <xsd:element name="SingleSignonProtocolProfile"  
305           type="xsd:anyURI" maxOccurs="unbounded" />  
306       </xsd:sequence>  
307       <xsd:attribute name="IssuerID" type="xsd:string"  
308         use="optional" />  
309     </xsd:extension>  
310   </xsd:complexContent>  
311 </xsd:complexType>
```



313

314 The element IDPDescriptor does not explicitly contain a SAML Artifact Source ID as specified in  
315 | Section 4.1.1.8 of [\[SAMLBind\]](#). In order to construct the Source ID, providers MUST use the  
316 following rules:

317 Each provider selects the identity provider's metadata attribute `providerID`,

318 The provider constructs the Source ID component of the artifact by taking the SHA-1 hash of the  
319 metadata attribute `providerID`, which results in a 20-byte binary value. Note that Source ID  
320 value, used to construct the artifact, is not encoded in hexadecimal.

### 321 **2.1.6.1 Element SoapEndpoint**

322 The element `SoapEndpoint` specifies the URL for the SAML SOAP Binding Service at the  
323 identity provider's site (as described in [\[SAMLBind\]](#) section 3.1.3).

### 324 **2.1.6.2 Element singleSignOnServiceURL**

325 The element `SingleSignOnServiceURL` specifies the URL for the identity provider's inter-site  
326 | transfer service (as described in [\[SAMLBind\]](#), sections 4.1.1.3 and 4.1.2.3)

### 327 **2.1.6.3 Element SingleSignonProtocolProfile**

328 The element `SingleSignonProtocolProfile` specifies the SAML browser profile(s)  
329 supported by the identity provider. Each instance of this element MUST have a value correspond  
330 | to one of the URIs specified in Section 4.1.1.1 or 4.1.2.1 of [\[SAMLBind\]](#).

### 331 **2.1.6.4 Attribute IssuerID**

332 The attribute `IssuerID` specifies the issuer ID of the assertion as described in Section 2.3 of  
333 | [\[SAMLBind\]](#). This is an optional attribute. When this attribute is not specified, a service provider  
334 MUST use the value of the attribute `providerID` to identify assertions issued by this Identity  
335 Provider (See section 2.1.5.1).

336

337 In order to maintain compatibility with the Liberty metadata specifications, it is RECOMMENDED  
338 that Identity Providers not include this attribute in their metadata specification. Instead, Identity  
339 Providers should use a URI for the value of their `saml:issuer` and specify the same URI as the  
340 value for the attribute `providerID` (See section 2.1.5.1). Specifying a URI for the value of  
341 | `saml:issuer` is consistent with the recommendations in Section 2.3 of [\[SAMLBind\]](#).

### 342 **2.1.7 Element SPDescriptor**

343 The element `SPDescriptor` is of type `SPDescriptorType`, which extends  
344 `providerDescriptorType` with the following element:

345 AssertionConsumerServiceURL [required]

346 SingleSignonProtocolProfile [required]

347

348 The schema fragment for `SPDescriptor` is:

349

```
<xsd:element name="SPDescriptor" type="SPDescriptorType" />
<xsd:complexType name="SPDescriptorType">
351 <xsd:complexContent>
352 <xsd:extension base="providerDescriptorType">
353 <xsd:sequence>
354
355 <xsd:element name="AssertionConsumerServiceURL" type="xsd:anyURI" />
```

356  
357  
358  
359  
360

```
<xsd:element name="SingleSignOnProtocolProfile" type="xsd:anyURI" />  
</xsd:sequence>  
</xsd:extension>  
</xsd:complexContent>  
</xsd:complexType>
```

### 361 **2.1.7.1 Element AssertionConsumerServiceURL**

362 The element AssertionConsumerServiceURL is a URL consisting of a host name and a path.  
363 For service providers that use SAML Browser/Artifact profile, the value of  
364 AssertionConsumerServiceURL MUST be set to the artifact receiver host name and path  
365 (section 4.1.1.5 of [\[SAMLBind\]](#)).

366

367 For service providers that use SAML Browser/POST profile, the value of  
368 AssertionConsumerServiceURL MUST be set to the assertion consumer host name and  
369 path (section 4.1.2.4 of [\[SAMLBind\]](#)).

370

371 Service providers that support both SAML Browser/Artifact and Browser/POST profiles SHOULD  
372 provide separate SPDescriptors, each having a different providerID and a different  
373 AssertionConsumerServiceURL. The two SPDescriptors MAY be encapsulated in a single  
374 EntityDescriptor.

### 375 **2.1.7.2 Element SingleSignonProtocolProfile**

376 The element SingleSignonProtocolProfile specifies the SAML browser profile(s)  
377 supported by the service provider. Its value MUST be one of the URNs specified in Section  
378 4.1.1.1 or 4.1.2.1 of [\[SAMLBind\]](#).

## 379 **2.2 Entity Descriptors**

380 An entity descriptor is a container for one or more provider descriptors for a single provider. It is  
381 permissible for an entity descriptor to contain more than one descriptor for the same provider. For  
382 example, a service provider that supports both Browser/Artifact and Browser/POST profiles may  
383 specify its metadata in two service provider descriptors and encapsulate both in a single entity  
384 descriptor. However, an entity descriptor MUST NOT contain descriptors from multiple providers.

### 385 **2.2.1 Element EntityDescriptor**

386 The element EntityDescriptor is a container for one or more provider descriptors. The  
387 schema fragment for EntityDescriptor is:

388  
389  
390  
391  
392  
393  
394

```
<xsd:element name="EntityDescriptor" type="entityDescriptorType" />  
<xsd:complexType name="entityDescriptorType" mixed="false">  
  <xsd:sequence maxOccurs="unbounded">  
    <xsd:element ref="SPDescriptor" minOccurs="0" />  
    <xsd:element ref="IDPDescriptor" minOccurs="0" />  
  </xsd:sequence>  
</xsd:complexType>
```

395

396

---

## 397 3 Examples

398 This section provides examples of metadata for Source (Identity Provider) and Destination  
399 (Service Provider).

### 400 3.1 Identity Provider

401 The example below illustrates the metadata for an Identity Provider that supports SAML  
402 Browser/Artifact and Browser/POST profiles. This metadata is not signed.

403

```
404 <EntityDescriptor>  
405 <IDPDescriptor  
406   providerID="http://www.IDP-bpa.net/SSOMechanism"  
407   validUntil="2003-10-11T00:33:32Z"  
408   protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol" >  
409   <ds:KeyInfo>  
410     <ds:X509Data>  
411       <ds:X509Certificate>  
412         MIICYjCCAiECAQAwCQYHKoZIzjgEAzAaMRgwFgYDVQQDEw9  
413         zaWdhYmEtc3BhLXNpZ24wHhcNMDMwMzE4MjIwMzU5WhcNMDM  
414         wNDE3MjIwMzU5WjAaMRgwFgYDVQQDEw9zaWdhYmEtc3BhLXNp  
415         .....  
416         .....  
417       </ds:X509Certificate>  
418     </ds:X509Data>  
419   </ds:KeyInfo>  
420   <Organization>  
421     <OrganizationName>Id Broker Corporation</OrganizationName>  
422     <OrganizationDisplayName>  
423       ID Broker Corp.  
424     </OrganizationDisplayName>  
425     <OrganizationURL>http://www.idbcorpusa.net</OrganizationURL>  
426   </Organization>  
427   <SoapEndPoint>https://www.IDP-bpa.net/serv/soaper</SoapEndPoint>  
428   <SingleSignOnServiceURL>  
429     https://www.IDP-bpa.net/serv/sso  
430   </SingleSignOnServiceURL>  
431   <SingleSignOnProtocolProfile>  
432     urn:oasis:names:tc:SAML:1.0:profiles:artifact  
433   </SingleSignOnProtocolProfile>  
434   <SingleSignOnProtocolProfile>  
435     urn:oasis:names:tc:SAML:1.0:profiles:browser-post  
436   </SingleSignOnProtocolProfile>  
437 </IDPDescriptor>  
438 </EntityDescriptor>
```

### 439 3.2 Service Provider

440 The example below illustrates the metadata for a Service Provider that only supports SAML  
441 Browser/Artifact profile. This metadata is not signed.

442

```
443 <EntityDescriptor>  
444 <SPDescriptor  
445   providerID="http://www.SP-bpa.net/BA-ResourceProviderMechanism"  
446   validUntil="2003-08-11T00:13:36Z"  
447   protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
```

```

448 <ds:KeyInfo>
449   <ds:X509Data>
450     <ds:X509Certificate>
451       MIICYjCCAIECAAwCQYHKoZIZjgEafAaMRgwF9YDVQQDEw9
452       zaWdhYmEtc3BhLyNpZ24wH2cNMDMwMzE4MjIwMzU5WhcNMDM
453       wlDE3MjIwMzU5WjAaMRgwFgYDVQQDEw9zaWdhYmEtc3BhLXNp
454       .....
455       .....
456     </ds:X509Certificate>
457   </ds:X509Data>
458 </ds:KeyInfo>
459 <Organization>
460   <OrganizationName>
461     Exemplary Resource Provider
462   </OrganizationName>
463   <OrganizationDisplayName>
464     Exemplart RP, Inc.
465   </OrganizationDisplayName>
466   <OrganizationURL>http://www.sp-bpa.net</OrganizationURL>
467 </Organization>
468 <AssertionConsumerServiceURL>
469   https://www.SP-bpa.net/serv/BA-SSOconsumer
470 </AssertionConsumerServiceURL >
471 <SingleSignOnProtocolProfile>
472   urn:oasis:names:tc:SAML:1.0:profiles:artifact
473 </SingleSignOnProtocolProfile>
474 </SPDescriptor>
475 </EntityDescriptor>

```

476

477 The example below illustrates the metadata for a Service Provider that supports SAML  
478 Browser/Artifact and Browser/POST profiles. This metadata is not signed.

479

480 In this example the Service Provider specifies two SPDescriptors, each having a unique  
481 providerID. This is to allow the Identity Providers to discern between the Browser/Artifact  
482 AssertionConusmerURL and the Browser/POST AssertionConsumerURL.

483

```

484 <EntityDescriptor>
485   <SPDescriptor
486     providerID="http://www.SP-bpa.net/BA-ResourceProviderMechanism"
487     validUntil="2003-08-11T00:13:36Z"
488     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
489     <ds:KeyInfo>
490       <ds:X509Data>
491         <ds:X509Certificate>
492           MIICYjCCAIECAAwCQYHKoZIZjgEafAaMRgwF9YDVQQDEw9
493           zaWdhYmEtc3BhLyNpZ24wH2cNMDMwMzE4MjIwMzU5WhcNMDM
494           wlDE3MjIwMzU5WjAaMRgwFgYDVQQDEw9zaWdhYmEtc3BhLXNp
495           .....
496           .....
497         </ds:X509Certificate>
498       </ds:X509Data>
499     </ds:KeyInfo>
500     <Organization>
501       <OrganizationName>
502         Exemplary Resource Provider
503       </OrganizationName>
504       <OrganizationDisplayName>
505         Exemplart RP, Inc.
506       </OrganizationDisplayName>

```

```

507         <OrganizationURL>http://www.sp-bpa.net</OrganizationURL>
508     </Organization>
509     <AssertionConsumerServiceURL>
510         https://www.SP-bpa.net/serv/BA-SSOconsumer
511     </AssertionConsumerServiceURL >
512     <SingleSignOnProtocolProfile>
513         urn:oasis:names:tc:SAML:1.0:profiles:artifact
514     </SingleSignOnProtocolProfile>
515 </SPDescriptor>
516 <SPDescriptor
517     providerID="http://www.SP-bpa.net/BPResourceProviderMechanism"
518     validUntil="2003-08-11T00:13:36Z"
519     protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
520     <ds:KeyInfo>
521         <ds:X509Data>
522             <ds:X509Certificate>
523                 MIICYjCCAiECAaAwCQYHKoZIzjgEAfAaMRgwF9YDVQQDEw9
524                 zaWdhYmEtc3BhLyNpZ24wH2cNMDMwMzE4MjIwMzU5WhcNMDM
525                 wlDE3MjIwMzU5WjAaMRgwFgYDVQQDEw9zaWdhYmEtc3BhLXNp
526                 .....
527                 .....
528             </ds:X509Certificate>
529         </ds:X509Data>
530     </ds:KeyInfo>
531     <Organization>
532         <OrganizationName>
533             Exemplary Resource Provider
534         </OrganizationName>
535         <OrganizationDisplayName>
536             Exemplart RP, Inc.
537         </OrganizationDisplayName>
538         <OrganizationURL>http://www.sp-bpa.net</OrganizationURL>
539     </Organization>
540     <AssertionConsumerServiceURL>
541         https://www.SP-bpa.net/serv/BP-SSOconsumer
542     </AssertionConsumerServiceURL >
543     <SingleSignOnProtocolProfile>
544         urn:oasis:names:tc:SAML:1.0:profiles:browser-post
545     </SingleSignOnProtocolProfile>
546 </SPDescriptor>
547 </EntityDescriptor>
548

```

---

## 549 4 Validating Metadata Over An Untrusted Channel

550 The text below is a copy of an email from Tim Moses. It is intended to generate discussion  
551 regarding metadata exchange.

552

### 553 4.1 Introduction

554 In the case that the metadata is not communicated between the source and destination over an  
555 authentic channel, the destination's copy of the metadata is not trustworthy. Yet, the  
556 trustworthiness of all subsequent communications between the source and destination depends  
557 upon the trustworthiness of the metadata. Therefore, a technique is required to validate the  
558 destination's copy of the metadata.

559

560 Metadata validation **MUST** be performed using an alternative authentic channel, such as a  
561 telephone call or facsimile transmission, company letterhead, business card or face-to-face  
562 exchange. In order to minimize the cost of communicating and validating metadata, it **MUST** be  
563 possible to validate metadata verbally and reliably in a telephone call.

564

### 565 4.2 Procedure

566 The following procedure is **RECOMMENDED**.

567

- 568 1. The source **SHALL** calculate a validation string for the metadata according to the procedure  
569 described below.
- 570 2. The source **SHALL** communicate the validation string over an authentic channel to the  
571 destination.
- 572 3. The source **SHALL** communicate the metadata over an untrusted channel to the destination.
- 573 4. The destination **SHALL** calculate the validation string for the received metadata according to  
574 the procedure described below.
- 575 5. If the validation string calculated by the destination is identical to the validation string received  
576 from the source, then the destination **MAY** install and rely upon the metadata, otherwise it  
577 **MUST NOT** rely on the metadata.

### 578 4.3 Validation string calculation

579 The validation string **SHALL** be calculated from the metadata by operating upon it with the SHA-1  
580 hash algorithm. The calculation shall be performed over the <EntityDescriptor> element, starting  
581 with the "<" character of the start tag and finishing with the ">" character of the end tag. The right-  
582 most 4 bytes of the resulting digest are discarded. The left-most 3 bits of each of the remaining  
583 16 bytes are discarded. The remaining sixteen 5-bit values are represented as alphanumeric  
584 characters according to the following table.

585 00000 > A

586 00001 > B

587 ... omitting I

588 11000 > Z

589 11001 > 3

590 11010 > 4

591 ...

592 11111 > 9

593

594 Finally, the alphanumeric string is divided into four sub-strings, each of four characters, and the  
595 sub-strings are separated by hyphens. For example: A4HY-8KLN-9T3M-7GK4

596

597 Note that hyphens are inserted to assist in dictation. Note also that the ambiguities between I  
598 and 1, O and 0 and Z and 2 are eliminated. Also, note that the ambiguity between upper and  
599 lower-case letters is eliminated.

---

## 5 References

- 600
- 601     **[RFC2119]**       S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,  
602                   <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 603     **[SAMLBind]**       P. Mishra (Editor), *Bindings and Profiles for the OASIS Security*  
604                   *Assertion Markup Language (SAML)*, Committee Specification 01,  
605                   available from <http://www.oasis-open.org/committees/security>, OASIS,  
606                   May 2002.
- 607     **[SAMLCore]**       P. Hallam-Baker, P., and E. Maler, (Editors), *Assertions and Protocol for*  
608                   *the OASIS Security Assertion Markup Language (SAML)*, Committee  
609                   Specification 01, available from [http://www.oasis-](http://www.oasis-open.org/committees/security)  
610                   [open.org/committees/security](http://www.oasis-open.org/committees/security), OASIS, May 2002.
- 611     **[SAMLReqs]**       D. Platt et al., *SAML Requirements and Use Cases*, OASIS, December  
612                   2001.
- 613     **[SAMLSecure]**     Security and Privacy Considerations for the OASIS Security Assertion  
614                   Markup Language (SAML), [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/cs-sstc-sec-consider-01.doc)  
615                   [open.org/committees/security/docs/cs-sstc-sec-consider-01.doc](http://www.oasis-open.org/committees/security/docs/cs-sstc-sec-consider-01.doc).
- 616     **[XMLSig]**         D. Eastlake et al., *XML-Signature Syntax and Processing*,  
617                   <http://www.w3.org/TR/xmlsig-core/>, World Wide Web Consortium.
- 618     **[libMD]**          P. Davis (Editor), *Liberty Alliance Project: Metadata description and*  
619                   *discovery protocols*, DRAFT version: 1.0-06, 13 April 2003.
- 620     **[LAProtSchema]** John D. Beatty, John Kemp, *Liberty Protocols and Schemas Specification*,  
621                   Draft Version 1.1-05, November 2002.
- 622
- 623     **[SAMLInterOp]** Prateek Mishra, *Proposed InterOp Scenario for SAML at Catalyst 2002*,  
624                   April 26, 2002, available at [http://lists.oasis-open.org/archives/saml-](http://lists.oasis-open.org/archives/saml-dev/200206/msg00209.html)  
625                   [dev/200206/msg00209.html](http://lists.oasis-open.org/archives/saml-dev/200206/msg00209.html)
- 626
- 627     **[SAMLMETA-XSD]** [sstc-saml-1.1-schema-meta-data-draft-03.xsd.xml](#)



628

---

## Appendix A. Revision History

Rev	Date	By Whom	What
00	2003-09-15	Jahan Moreh	Initial draft based on Draft 07 of SAML 1.1 Metadata specification.

629

630

## Appendix B. Issues list and resolution

Issue	Resolution
<p>Section 2.1 states: “The immediate child node of <code>EntityDescriptor</code> expects one or more of: <code>IDPDescriptor</code> (see Section 2.1.6) <code>SPDescriptor</code> (see Section 2.1.7)”</p> <p>Rob P. questions: Doesn’t “or more” conflict with the representation definition given previously?</p>	<p>Added text (just before section 2.1) to clarify as follows: Multiple <code>providerIDs</code> may be used to identify a single provider. This allows multiple unique identifiers for the provider. This is supported in the metadata schema by using multiple <code>SPDescriptors</code> (or multiple <code>IDPDescriptors</code>) in a metadata description, each associated with a single <code>providerID</code>.</p>
<p>Section 2.5.1. states that “In case the provider is an Identity Provider, the value of the <code>providerID</code> MUST be the same as the value of the <code>Issuer</code> attribute specified in Section 2.3 of <a href="#">[SAMLCore]</a>.”</p> <p>Rob P. comments that: “The issue here is that SAML “Issuer” is a string; NOT a URI. Thus, some SAML Issuer’s can’t be described in <code>providerID</code>.”</p>	<p>Added an optional attribute called <code>IssuerID</code> to the element <code>&lt;IDPDescriptor&gt;</code>. Added text recommending that the <code>IssuerID</code> should not be specified as it is expected that <code>saml:issuer</code> be a URI (per SAMLCore Section 2.3). In summary, a source (Identity Provider) could specify an issuer ID, but it need not (and it is recommended that it should not).</p>
<p>Section 2.1.5.2.Attribute <code>protocolSupportEnumeration</code></p> <p>The attribute <code>protocolSupportEnumeration</code> describes the protocol release type of the provider described by <code>providerID</code>. This attribute is of the type <code>NMTOKEN</code>, which allows for the enumeration of a set of SAML protocol releases which the provider MUST support. The data type of the tokens MUST be URNs and the value must be one of the following:</p> <p style="padding-left: 40px;">urn:oasis:names:tc:SAML:1.0 urn:oasis:names:tc:SAML:1.1</p>	<p>It should be the protocol URN. Note that there is another element called <code>SingleSignonProtocolProfile</code> (Section 2.1.6.3), which is (currently) either the Brower/Post URN or Browser/Artifact URN. The value of this element pretty much dictates the acceptable assertions. I.e., there is no need to identify assertion urns.</p>
<p>Rob P asks: It’s not clear to me whether this should identify the protocol version urn(s) or the assertion urn(s) or both.</p>	
<p>Section 2.1.5.8</p> <p>The element <code>ds:keyInfo</code> contains the provider’s public key in a format allowed by <a href="#">[XMLSig]</a>. The provider’s public key is used for</p>	<p>Resolution: Added text to clarify signing request/responses:</p> <p>Re: signing metadata: The public key that verifies the metadata is NOT in the metadata</p>

verifying signed assertions, requests, and responses.

document; added text to clarify this

Rob P. asks: What about requests and responses which can also be signed? For Requests, the provider is the Service Provider. What about the key used to sign the metadata (in ds:Signature below)?

#### Section 2.1.6

The provider constructs the Source ID component of the artifact by taking the SHA-1 hash of the metadata attribute `providerID`, which results in a 20-byte binary value. Note that Source ID value, used to construct the artifact, is not encoded in hexadecimal.

Possible Resolution: Based on the RECOMMENDATION in the 1.0 binding document (lines 561-569) the source ID is a sha-1 hash of the source's identification URL. I read this as consistent with the providerID. And, yes, Liberty does use the SHA-1 hash of providerID as the source ID. See liberty-architecture-bindings-profiles-v1.1.pdf line 863-869

Rob P. states: SAML defines the SourceID as the SHA-1 hash of the SOAP Binding Service URL (i.e. the SoapEndPoint). Section 2.1.5.1 above declares that the ProviderID of an IdP must be the Issuer.

632

633

634

635

636

---

637 **Appendix C. TO Dos**

638 Assign publicly available document location

---

## Appendix D. Notices

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

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

652 Copyright © OASIS Open 2003. *All Rights Reserved.*

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

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

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

669