OASIS

Extensible Resource Identifier (XRI) Resolution V2.0

Committee Draft 01, 14 March 2005

Document identifier:
> xri-resolution-V2.0-cd-01

Location:
> http://docs.oasis-open.org/xri/xri/V2.0

Editors:
> Gabe Wachob, Visa International <gwachob@visa.com>

Contributors:
> Drummond Reed, Cordance <drummond.reed@cordance.net>
> Dave McAlpin, Epok <dave.mcalpin@epok.net>
> Chetan Sabnis, Epok <chetan.sabnis@epok.net>
> Peter Davis, Neustar <peter.davis@neustar.biz>
> Mike Lindelsee, Visa International <mlindels@visa.com>

Abstract:
> This document defines both a standard and a trusted HTTP-based resolution mechanism for Extensible Resource Identifiers (XRIs), specifically XRIs conforming to *Extensible Resource Identifier (XRI) Syntax V2.0* **[XRISyntax]** or higher. For a non-normative introduction to the uses and features of XRIs, see the *Introduction to XRIs* **[XRIIntro]**. For the set of XRIs defined to provide metadata about other XRIs, see the *Extensible Resource Identifier (XRI) Metadata V2.0* **[XRIMetadata]**.

Status:
> This document was last revised or approved by the XRI Technical Committee on the above date. The level of approval is also listed above. Check the current location noted above for possible later revisions of this document. This document is updated periodically on no particular schedule.
> Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/xri.
> For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/xri/ipr.php.
> The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/xri.

Table of Contents

101 1 Introduction

102 1.1 XRI Resolution Framework

103 Extensible Resource Identifiers (XRIs) provide a uniform syntax for abstract identifiers as defined
104 in **[XRISyntax]**. Because XRIs may be used across a wide variety of communities and
105 applications (as database keys, filenames, directory keys, object IDs, XML IDs etc.), no single
106 resolution mechanism may prove appropriate for all XRIs. However, in the interest of promoting
107 interoperability, this specification defines a standard framework for XRI resolution consisting of
108 two parts:

109 • *Generic resolution* (section 2) is a simple, flexible resolution protocol for the authority
110 segment of an XRI that relies exclusively on HTTP/HTTPS as a transport.

111 • *Trusted resolution* (section 3) is an extension of the generic resolution protocol that uses
112 SAML assertions to create a chain of trust between the participating authorities.

113 Both of these protocols are extensible, as described in section 4. In addition, other XRI resolution
114 services or protocols may be defined by future versions of this specification or by other
115 specifications.

116 1.2 General Format and Reader's Guide

117 In order to make the technical material in this specification as clear and understandable as
118 possible, this document includes extensive examples, particularly of resolution requests and
119 responses. The examples themselves are non-normative. In addition, certain sections devoted
120 entirely to examples have been marked as non-normative.

121 Different readers, therefore, may wish to take different approaches depending on their context:

122 • Newcomers to XRIs and XRI resolution may wish to read the introductions and overview
123 sections and concentrate on the examples in order to quickly gain an understanding of XRI
124 resolution architecture.

125 • Technical reviewers may wish to concentrate on the normative text and skip the example
126 sections.

127 • Implementers may wish to follow the examples and refer to the normative text and
128 appendices as necessary for specific requirements.

129 1.3 Terminology and Notation

130 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",
131 "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this
132 document are to be interpreted as described in **[RFC2119]**. When these words are not capitalized
133 in this document, they are meant in their natural language sense.

134   `Examples look like this.`

135 `XML elements and attributes that appear in text look like this.`

136 Throughout this document, the XML namespace prefix saml: stands for the Security Assertion
137 Markup Language **[SAML]** namespace "urn:oasis:names:tc:SAML:2.0:assertion," regardless of
138 whether this namespace prefix is explicitly declared in the example or text. Similarly, the XML
139 namespace prefix ds: stands for the W3C Digital Signature **[XMLDSig]** Namespace
140 "http://www.w3.org/2000/09/xmldsig#", the namespace prefix xrid: stands for the namespace
141 "xri://$res*schema/XRIDescriptor*($v%2F2.0)", and the namespace prefix xs: stands for the

142 namespace "http://www.w3.org/2001/XMLSchema", again, whether or not they are explicitly
143 declared in the example or text. These namespace prefixes are summarized in Table 1.

saml	urn:oasis:names:tc:SAML:2.0:assertion
ds	http://www.w3.org/2000/09/xmldsig#
xrid	xri://$res*schema/XRIDescriptor*($v%2F2.0)
xs	http://www.w3.org/2001/XMLSchema

144 **Table 1: XML namespace prefixes used in this specification.**

145 Terms used in this document are defined in the glossary in Appendix C of **[XRISyntax]**.

146 2 Generic Resolution

147 2.1 Introduction

148 Generic XRI resolution is the process of determining a network endpoint associated with an XRI
149 in order to obtain metadata about the resource identified by the XRI, or to further interact with the
150 resource. This specification defines a generic resolution protocol based on HTTP/HTTPS as a
151 simple, general-purpose mechanism for accomplishing this task. Other XRI resolution services
152 may be defined by future versions of this specification or by other specifications.

153 Identifier management policies are defined on a community-by-community basis. With XRIs, the
154 resolution community is specified by the authority segment of the XRI. When a resolution
155 community chooses to create a new identifier authority, it SHOULD define a policy for assigning
156 and managing identifiers under this authority. Furthermore, it SHOULD define what resolution
157 protocol(s) can be used for resolving identifiers assigned by the authority.

158 2.1.1 Assumptions

159 The generic resolution protocol makes the following minimal assumptions about the XRIs being
160 resolved:

161 • The endpoints representing the top-level authority for any absolute XRI are identified
162   by the authority segment ("xri-authority" or "i-authority" productions) of the XRI as
163   defined in section 2.2.1 of **[XRISyntax]**.

164 • Only absolute XRIs can be resolved using this protocol. To resolve a relative XRI
165   reference, it must be converted into an absolute XRI using the procedure defined in
166   section 2.4 of **[XRISyntax]**.

167 • The XRI being resolved has been converted into URI-normal form, following the rules
168   in section 2.3.1 of **[XRISyntax]**.

169 • A resource represented by a single XRI may be accessed by multiple protocols at
170   multiple protocol endpoints. For example, it is possible that a resource represented
171   by a single XRI may be accessed through multiple HTTP URIs, or through both HTTP
172   and another network protocol. While only HTTP access to resources is defined by
173   this specification, an extension mechanism for specifying access via URIs in other
174   schemes is also defined.

175 • Each network endpoint associated with a resource identified by an XRI may present
176   a different subset, type, or representation of data or metadata associated with the
177   identified resource. For example, two separate HTTP URIs may be associated with a
178   single XRI, one for data access and the other for metadata access. This specification
179   allows XRI authorities to define multiple access types using extensible descriptor
180   fields based on content type and the semantics of the interaction.

181 2.1.2 Phases of Resolution

182 The generic resolution protocol is designed to be as simple and flexible as possible given the
183 assumptions above. Based on the structure of XRIs, it consists of two phases:

184 • Authority resolution

185 • Local access

186 Authority resolution is the process of finding the endpoint or endpoints that are authoritative for
187 access to resources under that authority's control, or of discovering further information about the
188 authority itself. In the case where the desired goal is access to a resource, the result of authority
189 resolution will be a list of local access endpoints, identified by one or more URIs, that support at

190 least one local access protocol. The calling application may then choose one of these endpoints
191 and access it using its choice of any supported local access protocol.
192 In the case where the goal of resolution is to discover more information about an authority, such
193 as XRI synonyms, public keys, or other XRI resolution metadata, this information will be returned
194 by the authority resolution process itself.
195 Figure 1 illustrates the two main phases of XRI resolution – authority resolution and local access:
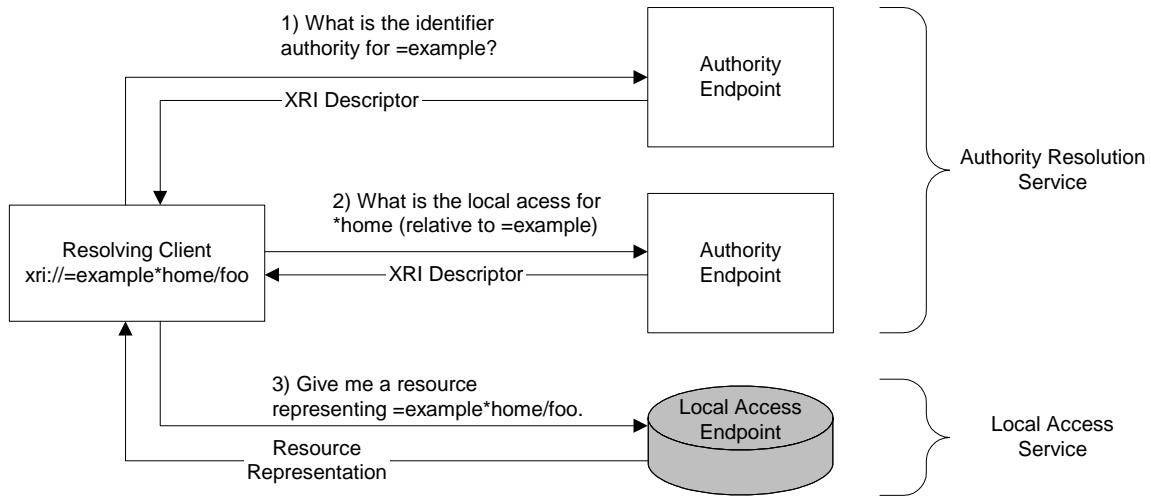
196
197 **Figure 1: Phases of Resolution**

2.1.3 XRI vs. IRI Authorities

198

199 As described in section 2.2.1 of **[XRISyntax]**, XRI authorities and IRI authorities have different
200 syntactic structures, partially due to the higher level of abstraction represented by XRI authorities.
201 For this reason, XRI authorities are resolved to authority descriptor documents one sub-segment
202 at a time, as described in section 2.2. IRI authorities, since they are based on DNS names or IP
203 addresses, are resolved into an authority descriptor through a special HTTP(S) request based on
204 the DNS name or IP address identified by the IRI authority segment.

2.1.4 XRI Metadata Reserved for XRI Resolution

205

206 As defined in section 2.2.1.2 of **[XRISyntax]**, the GCS symbol "$" is reserved for special
207 identifiers assigned by XRI TC specifications, other OASIS specifications, or other standards
208 bodies. (See also **[XRIMetadata]**.) Within the "$" namespace, the identifier "$res" is reserved for
209 identifiers assigned by this XRI resolution specification. Table 2 summarizes these identifiers.
210

Identifier	Use	See Section
xri://$res*schema	XML namespace for XRI resolution schema	2.2.2
xri://$res*auth.res	Namespace for authority resolution protocol types	2.2.4
xri://$res*local.access	Namespace for local access protocol types	2.4.1
xri://$res*trusted	Namespace for trust mechanisms	2.2.2 and 3

211 **Table 2: Special identifiers reserved for XRI resolution.**

2.2 XRI Authority Resolution

2.2.1 Overview

212

213

214 XRI authority resolution is an iterative process that resolves the qualified sub-segments within the
215 XRI authority segment from left to right. A qualified sub-segment is a sub-segment as defined by
216 the productions whose names start with "xri-subseg" in section 2.2.3 of **[XRISyntax]** *including the*
217 *leading syntactic delimiter* ("*" or "!"). Note that a qualified sub-segment always includes the
218 leading syntatic delimiter even if it was optionally omitted in the original XRI (see section 2.2.3 of
219 **[XRISyntax]**).

220 The first (or leftmost) component of the XRI authority segment specifies the root of the identifier
221 community. In XRI syntax this can be either:

222 • a global context symbol as defined by section 2.2.1.2 of **[XRISyntax]**, or

223 • a cross-reference as defined by section 2.2.2 of **[XRISyntax]**.

224 The qualified sub-segment immediately to the right of the root is resolved in the context of the
225 root, and all subsequent sub-segments are resolved in the the context of the sub-segment
226 immediately to their left.

227 Each sub-segment is resolved to a corresponding XRI Descriptor (often abbreviated as "XRID"),
228 an XML document that specifies one or more network endpoints (in the case of authority
229 resolution defined here, HTTP or HTTPS URIs) that answer XRI resolution requests. As
230 resolution proceeds, the XRI resolver is building a "chain" (i.e., an ordered list) of XRID
231 documents. Resolution is complete when the resolver has followed the chain of XRIDs for all sub-
232 segments in the XRI authority segment. Figure 2 and Figure 3 below depict this resolution
233 process for the XRI authority "@a*b*c":
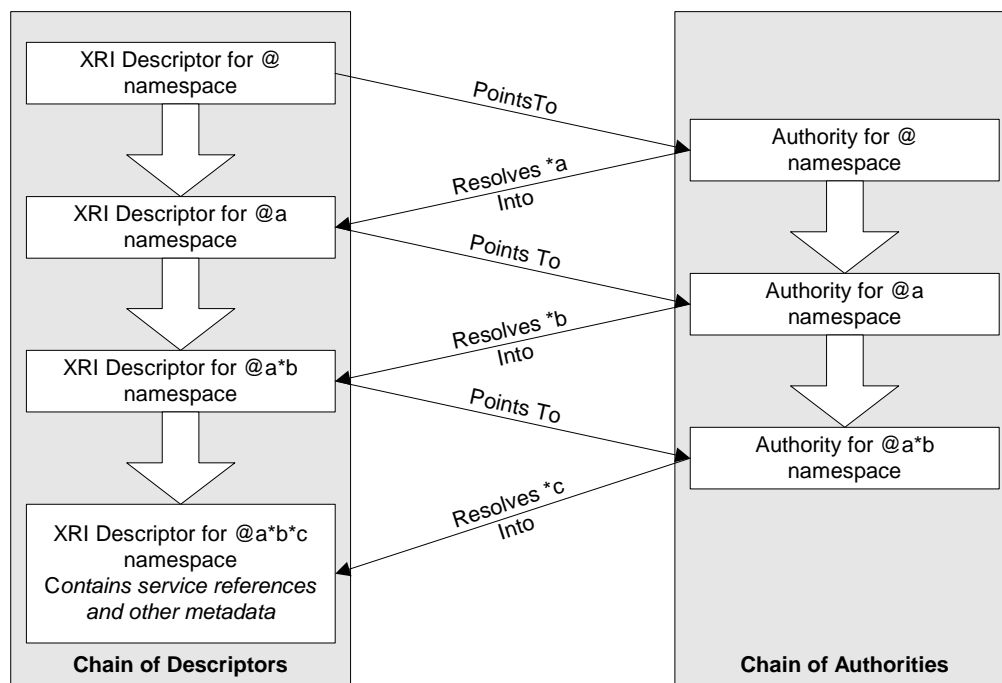
234

235 **Figure 2: XRI Descriptors, XRI Authorities and Authority Sub-segments for @a*b*c**

Resolving Client | @ Authority | @a Authority | @a*b Authority

Resolve(*a)

XRIDescriptor(@a)

Resolve(*b)

XRIDescriptor(@a*b)

Resolve(*c)

XRIDescriptor(@a*b*c)
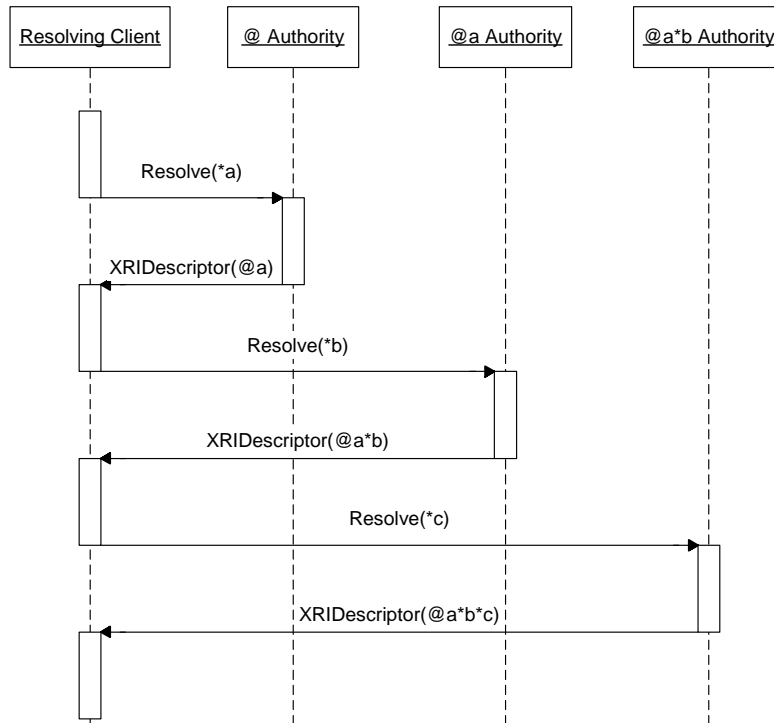
236

237 **Figure 3: XRI Authority Resolution Sequence Diagram**

238 Any resolution request may ask for resolution of more than one sub-segment—a feature called
239 *lookahead* resolution. If lookahead resolution is used, each response may contain one or more
240 XRI Descriptors inside a XML container document. The number of sub-segments resolved in one
241 resolution request depends on:

242 • How many sub-segments the resolving client presents to a responding XRI Authority for
243 lookahead resolution; and

244 • The configuration, policy, and state of the responding XRI Authority (e.g. previously
245 cached requests).

246 Each XRI Descriptor in the chain contains one or more of four basic types of information about
247 the XRI authority it describes:

248 • URIs describing network endpoints for XRI authority resolution services;

249 • URIs describing network endpoints for local access services;

250 • XRI synonyms (equivalent XRIs) for the resolved sub-segment.

251 • Additional information about the XRI authority included using the extension mechanisms
252 described in section 4.1.

253 All four types of information defined by this document—authority resolution services, local access
254 endpoints, XRI synonyms, or additional information—may be available at each step of resolution.
255 For example, the XRI authority identifier "@a*b*c" may be the prefix to another XRI authority with
256 the XRI "@a*b*c*d". "@a*b*c" may also be a local access endpoint itself, in which case its XRI
257 Descriptor will contain references to local access services. "@a*b*c" may also present synonyms
258 in its XRI Descriptor. One important use of synonyms is to map XRIs into "persistent XRIs". For
259 example, "@a*b*c" may have a persistent XRI synonym such as "xri://@!1000!2!3", which may
260 also be included in the XRID to indicate it is an equivalent persistent XRI.

2.2.2 XRI Descriptors

261

262 To provide a straightforward, flexible resolution mechanism, XRI authority endpoints are
263 described using a simple, flexible XML document, called an XRI Descriptor (abbreviated "XRID").
264 While this specification defines only XRID elements necessary to support delegated resolution
265 and access of XRI-identified authorities and resources, an XRID can easily be extended to
266 publish any form of metadata about the described authority.

267 The formal XML Schema definition of an XRI Descriptor is provided in Appendix B. The following
268 example instance document illustrates the fields defined in this schema:

```
269    <XRIDescriptors xmlns="xri://$res*schema/XRIDescriptor*($v%2F2.0)">
270        <XRIDescriptor xrid:id="first">
271            <Resolved>*foo</Resolved>
272             <AuthorityID>urn:uuid:c9f812f3-6544-4e3c-874e-
273    d3ae79f4ef7b</AuthorityID>
274            <Expires>2005-05-30T09:30:10Z</Expires>
275            <Authority>
276                <AuthorityID>urn:uuid:f0502a17-4503-4463-8516-
277    f1225b330e4d</AuthorityID>
278                <Type>xri://$res*auth.res/XRIA</Type>
279                <URI>http://xri.example.com</URI>
280                <URI>https://xri.example.com</URI>
281            </Authority>
282            <Service>
283                <Type>xri://$res*local.access/X2R</Type>
284                <URI>http://xri.example.com</URI>
285                <MediaType>application/rdf+xml</MediaType>
286            </Service>
287            <Service>
288                <Type>xri://$res*local.access/X2R</Type>
289                <URI>http://pictures.xri.example.com</URI>
290                 <MediaType>image/jpeg</MediaType>
291            </Service>
292            <Synonyms>
293                <Internal>xri://@!1!2!3</Internal>
294                <External>xri://@!4!5!6</External>
295            </Synonyms>
296            <TrustMechanism>xri://$res*trusted/None</TrustMechanism>
297        </XRIDescriptor>
298        Other XRIDescriptor elements here
299    </XRIDescriptors>
```

300 All schema elements in the basic XML Descriptor are in the XML namespace
301 "xri://$res*schema/XRIDescriptor*($v%2F2.0)". The following are the elements and attributes that
302 comprise the XRIDescriptor document type (all XPATHs are relative to the enclosing
303 `xrid:XRIDescriptors` document element):

304 **xrid:XRIDescriptor**

305   1 or more within the `xrid:XRIDescriptors` container. Has an "xrid:id" attribute to
306   uniquely identify this element within the containing `xrid:XRIDescriptors` document.

307 **xrid:XRIDescriptor/xrid:Resolved**

308   1 per `xrid:XRIDescriptor`. Required. Expresses the qualified sub-segment whose
309   resolution results in this `xrid:XRIDescriptor` element.

310 **xrid:XRIDescriptor/xrid:AuthorityID**

311   1 per `xrid:XRIDescriptor`. Required. A unique identifier of type `xs:anyURI` for the
312   authority that produced this XRI Descriptor. The value of this element MUST be such that
313   there is negligible probability that the same value will be assigned as an identifier to any

314	other authority. Note that the authority identified by this element is the *describing*
315	authority (the producer of the current XRID), not the authority *described* by the XRID. The
316	latter is specified in the
317	`xrid:XRIDescriptor/xrid:Authority/xrid:AuthorityID` element (see below).

318 xrid:XRIDescriptor/xrid:Expires

319	0 or 1. The date/time, in the form of `xs:dateTime`, after which this XRI Descriptor
320	cannot be relied upon. To promote interoperability, this date/time value SHOULD use the
321	UTC "Z" time zone and SHOULD NOT use fractional seconds. A resolver using this XRI
322	Descriptor MUST NOT use the XRI Descriptor after the time stated here. A resolver MAY
323	discard this Descriptor before the time indicated in this result. If the HTTP transport
324	caching semantics specify an expiry time that is earlier than the time expressed in this
325	attribute, then a resolver MUST NOT use this XRI Descriptor after the expiry time
326	declared in the HTTP headers per section 13.2 of **[RFC2616]**.

327 xrid:XRIDescriptor/xrid:Authority

328	0 or more. Describes an authority resolution service associated with the resolved
329	identifier. If there are additional sub-segments in the authority segment of the XRI being
330	resolved, they can be resolved at this service endpoint.

331 xrid:XRIDescriptor/xrid:Authority/xrid:AuthorityID

332	1 per `xrid:Authority` element. Required. The unique identifier of the authority
333	*described* by this `xrid:Authority` element, of type `xs:anyURI`. The value of this
334	element MUST be such that there is negligible probability that the same value will be
335	assigned as an identifier to any other authority. This element is correlated to the
336	`xrid:XRIDescriptor/xrid:AuthorityID` element described above. When the
337	authority *described* by this `xrid:Authority` element responds to resolution requests, it
338	will include this AuthorityID in the `xrid:XRIDescriptor/xrid:Authority` element
339	of its response. This element is particularly important in trusted resolution (see section 3).

340 xrid:XRIDescriptor/xrid:Authority/xrid:Type

341	0 or 1 per `xrid:Authority` element. Indicates the type of authority resolution service
342	described by the parent xrid:Authority element. This specification defines one authority
343	resolution service: "xri://$res*auth.res/XRIA" (XRI Authority resolution as described in
344	section 2.2.4). This is the default value if this element is not present.

345 xrid:XRIDescriptor/xrid:Authority/xrid:URI

346	1 or more per `xrid:Authority` element. Indicates the transport-level URI where the
347	authority resolution service described may be accessed. For the services defined in this
348	document, this URI MUST be an HTTP or HTTPS URI. Future versions of this
349	specification (or other specifications) may allow other transport protocols. Each URI
350	element has an optional attribute called "trusted" that indicates whether or not the
351	particular service endpoint provides trusted resolution (section 3). The trust mechanism is
352	described using the xrid:TrustMechanism element (below).

353 xrid:XRIDescriptor/xrid:Service

| 354 | 0 or more. Describes a local access service endpoint provided by the described authority. |

355 xrid:XRIDescriptor/xrid:Service/xrid:Type

356	0 or 1 per `xrid:Service` element. Indicates the type of local service being described.
357	This specification defines one service: "xri://$res*local.access/X2R" (the X2R local
358	access service as defined in section 2.4.2). This is the default value if this element is not
359	present.

360 xrid:XRIDescriptor/xrid:Service/xrid:URI

361   1 or more per `xrid:Service` element. Indicates the transport-level URI where the
362   service described may be accessed. For the X2R local access service defined in section
363   2.4.2, this URI MUST be an HTTP or HTTPS URI. Other services may use other
364   transport protocols.

xrid:XRIDescriptor/xrid:Service/xrid:MediaType

366   0 or more per `xrid:Service` element. The media type of content available at this
367   service. If this element is not present, then a processor of the Descriptor SHOULD NOT
368   make any assumption about the type of data available at this endpoint. The value of this
369   element must be of the form of a media type defined in **[RFC2046]**. This element may
370   appear multiple times to indicate all the media types available through this local access
371   service.

xrid:XRIDescriptor/xrid:Synonyms

373   0 or 1. Contains statements about the equivalence of the resolved identifier to other XRIs.

xrid:XRIDescriptor/xrid:Synonyms/xrid:Internal

375   0 or more. Represents another XRI assigned to the described authority by the current
376   describing authority. Must be an absolute XRI ("absolute-xri" in section 2.2 of
377   **[XRISyntax]**). An internal synonym may be used, for example, to assert that a XRI
378   authority known by a reassignable XRI may also be known by one or more persistent
379   XRIs, or by a different reassignable XRI than the one being resolved. Both cases may be
380   particularly useful in populating or querying a cache, since resolution of an internal
381   synonym will typically result in an XRID containing the same information as the current
382   XRID.

xrid:XRIDescriptor/xrid:Synonyms/xrid:External

384   0 or more. Represents another XRI assigned to the described authority by an authority
385   other than the current describing authority. Must be an absolute XRI ("absolute-xri" in
386   section 2.2 of **[XRISyntax]**). Resolution of an external synonym will typically result in an
387   XRID containing information different from that available in the current XRID. External
388   synonyms are used, for example, in XRI redirects, described in Section 2.2.7. They can
389   also be used to identify alternative sources of local access descriptors if those in the
390   current XRID do not satisfy the needs of the client.

xrid:XRIDescriptor/xrid:TrustMechanism

392   0 or 1. Identifies the mechanism for trusted resolution associated with this XRID. This
393   specification defines two values: "xri://$res*trusted/XRITrusted" (for Trusted Resolution
394   as described in section 3) and "xri://$res*trusted/None" (for generic resolution as
395   described here in section 2). If this element does not appear, the default value is
396   "xri://$res*trusted/None".

397   XRI Descriptor documents have an "open schema" that allows other elements and attributes from
398   other namespaces to be added throughout. These points of extensibility can be used to deploy
399   new identifier authority resolution schemes, new local access resolution schemes, additional XRI
400   synonym metadata, or other metadata about the described authority. See section 4.1 for more
401   about XRID extensibility.

402   See section 3.3.1 for information about additional XRI Descriptor elements defined for trusted
403   resolution.

2.2.3 Starting the Chain of XRI Descriptors with the Root XRID

405   With an XRI authority, the first sub-segment corresponding to the community root may be either a
406   global context symbol (GCS) character or top-level cross-reference as specified in section 2.2.1.1
407   of **[XRISyntax]**. In either case, the corresponding root XRID (or its equivalent) specifies the top-
408   level authority resolution endpoints for that community. The root XRID, or its location, is known *a*
409   *priori* and is part of the configuration of a resolver, similar to the specification of root DNS servers

410 in a DNS resolver. (Note that is not strictly necessary to publish this information in an XRID—it
411 may be supplied in any format that enables configuration of the XRI resolvers in the community—
412 but providing an XRID at a known location simplifies the process.)

413 If the first sub-segment of an XRI authority is a GCS character and the following sub-segment
414 does not begin with a "*" (indicating a reassignable sub-segment) or a "!" (indicating a persistent
415 sub-segment), then a "*" is implied and must be added when constructing the qualified sub-
416 segment. Table 3 and Table 4 illustrate the differences between parsing a reassignable sub-
417 segment following a GCS character and parsing a cross-reference, respectively.

XRI	xri://@example*internal/foo
XRI Authority	@example*internal
Community Root Authority	@
First Qualified Sub-Segment Resolved	*example

418 **Table 3: Parsing the first sub-segment of an XRI that begins with a global context symbol.**

XRI	xri://(http://www.example.com)*internal/foo
XRI Authority	(http://www.example.com)*internal
Community Root Authority	(http://www.example.com)
First Qualified Sub-Segment Resolved	*internal

419 **Table 4: Parsing the first sub-segment of an XRI that begins with a cross-reference.**

420 2.2.4 Default HTTP(S)-based Authority Resolution Service

421 This section defines the default authority resolution service for generic XRI resolution. When
422 explicitly declared, it uses the `xrid:XRIDescriptor/xrid:Authority/xrid:Type` element
423 value "xri://$res*auth.res/XRIA".

424 The generic (and trusted) XRI authority resolution service allows a client to request resolution of
425 multiple authority sub-segments in one transaction (lookahead resolution). If a client makes such
426 a request, the responding authority MAY perform the additional lookahead resolution steps
427 requested. In this case the responding authority acts as a client to the other authorities that need
428 to be queried for the lookahead segments. Alternatively, it may retrieve Descriptors only from its
429 local cache until it reaches a sub-segment whose XRID is not locally cached, or it may simply
430 lookahead only as far as it is authoritative. Any of these behaviors are reasonable, as are others
431 not described here.

432 If an authority performs any lookahead resolution, it MUST return an ordered list of
433 `xrid:XRIDescriptor` elements in an `xrid:XRIDescriptors` document. Each XRI
434 Descriptor MUST correspond to a sub-segment resolved by the authority on behalf of the
435 resolving client. The list of `xrid:XRIDescriptor` elements in the `xrid:XRIDescriptors`
436 document MUST appear in the same order as the sub-segments in the original request. The
437 responding authority MAY resolve fewer sub-segments than requested by the client. The
438 responding authority is under no obligation to resolve more than the first sub-segment (for which
439 it is, by definition, authoritative).
440
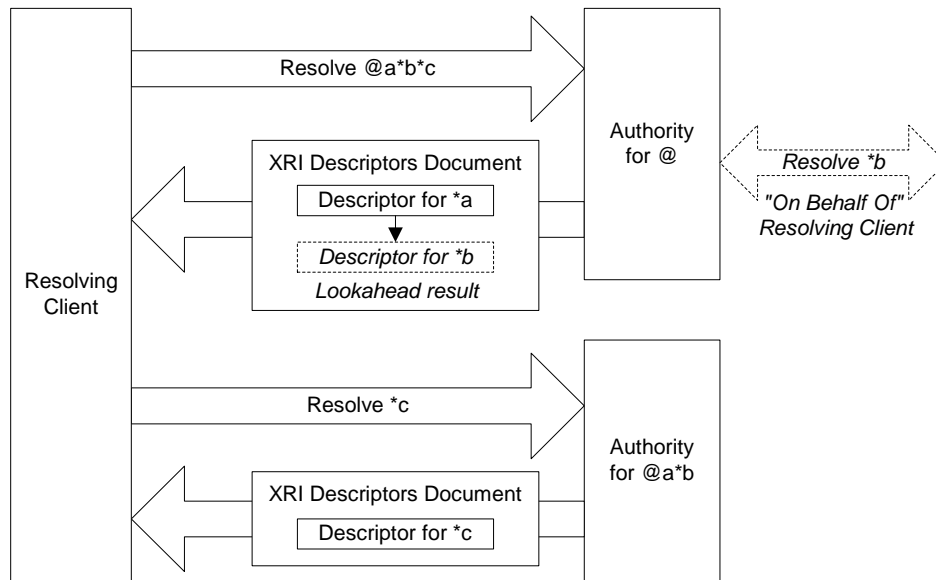
Figure 4: Lookahead Resolution

441
442

443 Figure 4 illustrates a resolving client requesting lookahead resolution for the XRI authority
444 "@a*b*c". The "@" authority is willing to resolve "@a*b" on behalf of the resolving client. The "@"
445 authority can accomplish this either by acting as a resolving client itself, or by examining a cache
446 it may have built through previous resolutions. In this example, the "@" authority it is only willing
447 or able to resolve the descriptor for "*a" (for which it is authoritative) plus "@a*b". Therefore the
448 resolving client must resolve "*c" itself. The resolving client will know the "@" authority only
449 resolved two segments (*a and *b) because it only returned two XRI Descriptors corresponding to
450 those two sub-segments.

451 If the responding authority does not resolve the entire set of sub-segments presented, the
452 resolving client MUST continue the authority resolution process itself. At any stage, however, the
453 resolving client MAY request that the next authority resolve any additional unresolved sub-
454 segments. For example, in Figure 4, if the "@" authority had refused to do any lookahead, the
455 resolving client could have asked the "@a" authority to resolve the unresolved "*b*c" portion of
456 the XRI authority segment.

2.2.4.1 Determining the URI for the Next Resolution Step

458 Before each authority resolution step is performed, a URI must be constructed for the next
459 HTTP(S) request. This URI establishes the context of that authority. Initially the current context is
460 the root authority, and the current context shifts to a new authority each time a resolution step is
461 performed. After a lookahead resolution request, the current context is the last authority whose
462 sub-segment was resolved by the authority performing the lookahead request.

463 This "Next Authority URI" is constructed from two strings:

464 • The contents of the `xrid:XRIDescriptor/xrid:Authority/xrid:URI` element
465 extracted from the XRI Descriptor corresponding to the current context, and

466 • The next qualified sub-segment to be resolved. (Note that this sub-segment must begin
467 with an XRI syntax delimiter, i.e., "*" or "!" —see section 2.2.6.)

468 If the path portion of the first URI does not end with a forward slash ("/"), one must be appended
469 before proceeding. Then the URI-normal form (section 2.3.1 of **[XRISyntax]**) of the next qualified
470 sub-segment being resolved is appended to the path portion of this URI. For example, when
471 resolving the "c" sub-segment of "xri://@a*b*c*d", if the XRI Authority URI resulting from the
472 resolution of "xri://@a*b" is "http://example.com/xri-authority/", then the Next Authority URI is the
473 concatenation of "http://example.com/xri-authority/" with "*c", yielding "http://example.com/xri-

474　authority/*c". An HTTP GET request is made to this URI, and the XRID for the context
475　"xri://@a*b*c" is retrieved.

476　For lookahead resolution (Figure 4), any portion of the remaining XRI authority segment may be
477　appended, not just the first sub-segment. For example, if the resolving client wanted to resolve
478　"*c*d", it would append this entire string to "http://example.com/xri-authority/", yielding
479　"http://example.com/xri-authority/*c*d".

480　Construction of the Next Authority URI is more formally described in this pseudo-code for
481　resolving a "sub-segment-list" via an HTTP URI called "xa-uri":

```
482    xa-uri = xri-authority-uri
483
484    if (path portion of xa-uri doesn't end in "/"):
485        append "/" to path portion of xa-uri
486
487    if (sub-segment-list isn't preceded with "*" or "!" separator):
488        prepend "*" to sub-segment-list
489
490    append uri-escape(sub-segment-list) to path portion of xa-uri
```

491 2.2.4.2 Making HTTP(S) Resolution Requests

492　Once the Next Authority URI is constructed, an HTTP or HTTPS GET request is made using this
493　URI. Each GET request results in either a 2XX or 304 HTTP response. The HTTP request
494　SHOULD contain an Accept header with the value of "application/xrid+xml". See section
495　3.3.3 for a different value that may appear in the Accept header during trusted resolution.

496　The ultimate HTTP/HTTPS response for a successful resolution MUST contain either: a) a 2XX
497　response with an XRI Descriptors document containing a list of one or more
498　xrid:XRIDescriptor elements, or b) a 304 response signifying that the cached version on the
499　client is still valid (depending on the client's HTTP request). HTTP caching semantics should be
500　leveraged to the greatest extent possible to maintain the efficiency and scalability of the HTTP-
501　based resolution system. The recommended use of HTTP caching headers is described in more
502　detail in section 2.5.1.

503　Any ultimate response besides an HTTP 2XX or 304 SHOULD be considered an error in the
504　resolution process. There is no restriction on intermediate redirects (i.e., 3XX result codes) or
505　other result codes (e.g., a 100 HTTP response) that eventually result in a 2XX or 304 response
506　through normal operation of **[RFC2616]**. Ultimately, the content of a successful response will be
507　an XRI Descriptors document containing one or more xrid:XRIDescriptor elements for the
508　qualified sub-segment(s) being resolved.

509　If there are no more sub-segments, the final context (as described by the final XRID retrieved)
510　can be used for local access services as described in section 2.4, or to obtain synonyms or other
511　metadata about the final authority.

512 2.2.4.3 Proxied Resolution

513　In some cases it may be desirable for a server to do fully proxied XRI resolution on behalf of a
514　client. While this is very similar to lookahead resolution, a lookahead resolution request is always
515　sent to the first sub-segment's authority. A proxied resolution request, in contrast, may be sent to
516　any XRI proxy server that will accept the request.

517　The proxy resolution service is very simple: an HTTP GET is performed on a URI constructed by
518　concatenating the base URI for the proxy resolution service and the XRI authority segment for
519　which proxy resolution is being requested. As with standard resolution, this XRI authority segment
520　MUST be in URI-normal form. Additionally, if the base proxy URI does not contain a trailing
521　forward slash ("/"), one MUST be inserted between the base URI and the XRI authority segment.
522　The proxy answering this request MUST perform XRI authority resolution as specified in this
523　document and MUST return either an XRI Descriptors document containing a entire chain of

524 `xrid:XRIDescriptor` elements for the segments of the authority it resolves, or an HTTP error
525 code as described in section 2.2.4.4.

526 Note that because a proxy is not associated with any specific authority, proxied resolution
527 requests MUST be composed of authority segments starting with a GCS character or a cross-
528 reference identifying a community root authority. In addition, a proxy resolver MUST return an
529 XRI Descriptor chain that begins with an XRID describing the community root authority. If the
530 community root authority does not publish an XRID itself, a proxy MUST construct one from the
531 equivalent information published by the community root authority.

532 The following example illustrates a proxied resolution request for "xri://=example*home*base". It
533 assumes that the URI for a local proxy server is "http://proxy.example.com/xri-proxy". First the
534 following HTTP GET request is made to "proxy.example.com":

```
535     GET /xri-proxy/=example*home*base HTTP/1.1
536     <other HTTP headers>
```

537 The proxy resolver then performs authority resolution, behaving as a resolving client as described
538 in section 1. After completing this resolution process, the proxy resolver might produce the
539 following HTTP response:

```
540     200 OK HTTP/1.1
541     Content-Type: application/xrid+xml
542     Expires: Fri, 7 Nov 2003 19:43:31 GMT
543     <other HTTP headers>
544
545     <XRIDescriptors xmlns="...">
546     <XRIDescriptor>
547       <Resolved>=</Resolved>
548       …
549     </XRIDescriptor>
550     <XRIDescriptor>
551       <Resolved>*example</Resolved>
552       …
553     </XRIDescriptor>
554     <XRIDescriptor>
555       <Resolved>*home</Resolved>
556       …
557     </XRIDescriptor>
558     <XRIDescriptor>
559       <Resolved>*base</Resolved>
560       <AuthorityID>
561         urn:uuid:C9FBEE76-9438-11D9-8BDE-F66BAD1E3F3A
562       </AuthorityID>
563     <Service>
564       <Type>
565         xri://$res*local.access/X2R
566       </Type>
567         <URI>
568           http://xri.other.example.com/xri-local/base/
569         </URI>
570         <URI>
571           https://xri.other.example.com/xri-local/base/
572         </URI>
573       </Service>
574       …
575     </XRIDescriptor>
576
577     </XRIDescriptors>
```

578   The resolving client can then parse this XRI Descriptor and extract the Local Access element
579   from the last XRI Descriptor element.

580   Note that proxy resolvers are uniquely positioned to take advantage of caching and SHOULD use
581   it to resolve the same authority sub-segments for multiple clients.

582   A proxy resolution service does not provide a complete XRI-to-resource mapping service. The
583   client must still parse the returned XRID and invoke an appropriate local access service, if
584   desired. For the default X2R local access protocol, a complete mapping service could be defined
585   by sending the proxy server the complete XRI in XRI-normal form and having it return an HTTP
586   redirect to the local access URI. Alternatively, the proxy server could return the resource directly
587   by performing the local access request itself. Neither method, however, is prescribed or defined
588   by this document.

2.2.4.4 Errors During Proxied and Lookahead Authority Resolution

590   Proxies and lookahead resolvers MUST "pass through" to the resolving client any HTTP error
591   codes resulting from resolution if the proxy or lookahead resolver cannot proceed with resolution
592   due to an HTTP error condition. For example, if, during resolution on behalf of a client, a proxy is
593   returned a 404 error code by an authoritative server, it must return that 404 code to its client.

594   Upon encountering an HTTP error code that halts the proxy or lookahead resolver's ability to
595   complete resolution, the proxy or lookahead resolver MUST return an xrid:XRIDescriptors

596 document in the body of the HTTP error response. This `xrid:XRIDescriptors` document
597 MUST contain the list of XRI Descriptor elements corresponding to the sub-segments
598 successfully resolved or retrieved from cache. For example, if a proxy is asked to resolve
599 @a*b*c, and successfully resolves @a*b, but receives a HTTP 404 on resolving *c, it will return
600 an HTTP 404 response to its client that include `xrid:XRIDescriptor` elements for @, *a, and
601 *b. In this way, the resolver indicates to the resolving client that *c is the sub-segment causing the
602 404 response.

603 This use of error codes, while slightly unusual, conforms to the requirements of **[RFC2616]**,
604 specifically sections 10.4 and 10.5, which state that "the server SHOULD include an entity
605 containing an explanation of the error situation." The combination of the error code and the list of
606 successfully resolved `xrid:XRIDescriptor` elements explains to the client exactly which sub-
607 segment caused the error. This should save both the client and the authority returning the error
608 code an extra HTTP request/response cycle.

609 Even when given an HTTP error response, resolving clients SHOULD consider the
610 `xrid:XRIDescriptor` elements returned in the content of the HTTP response as valid
611 cacheable responses (if the client does caching). All other rules about XRI Descriptors, including
612 those specified in Section 3 for trusted resolution, also apply.

613 2.2.5 Examples (Non-Normative)

614 2.2.5.1 Authority Resolution without Lookahead

615 In the following example, the authority portion of an XRI is resolved without lookahead. That is,
616 for each resolution step, the resolving client requests resolution of only one authority sub-
617 segment of the following XRI:

```
618        xri://=example*home*base/foo*bar
```

619 This example assumes that the URI for the "=" global context symbol is
620 http://equals.example.org/xri-resolve, found in
621 `xrid:XRIDescriptor/xrid:Authority/xrid:URI` of the XRID for this community.

622 Resolving "=example"

623 The following HTTP request is made to "equals.example.org":

```
624        GET /xri-resolve/*example HTTP/1.1
625        If-Modified-Since: Fri, 31 Oct 2003 19:43:31 GMT
626        Accept: application/xrid+xml
627        <other HTTP headers>
```

628 The following HTTP response is received from "equals.example.org" (the content has changed
629 since "Fri, 31 Oct 2003 19:43:31 GMT", the value specified in the the "If-Modified-Since" header):

```
630    200 OK HTTP/1.1
631    Content-Type: application/xrid+xml
632    Expires: Fri, 7 Nov 2003 19:43:31 GMT
633    <other HTTP headers>
634
635    <XRIDescriptors xmlns="…">
636    <XRIDescriptor>
637    <Resolved>*example</Resolved>
638    <AuthorityID>
639    urn:uuid:2BA56CDE-9438-11D9-8BDE-F66BAD1E3F3A
640    </AuthorityID>
641    <Authority>
642    <AuthorityID>
643    urn:uuid:925B458F-5907-7654-C3F9-BE3D8912BA73
644    </AuthorityID>
645    <URI>
646    http://xri.example.com/xri-resolve/
647    </URI>
648    </Authority>
649    <Service>…</Service>
650    </XRIDescriptor>
651    </XRIDescriptors>
```

Resolving "=example*home"

653 Appending the next qualified sub-segment "*home" to the URI "http://xri.example.com/xri-resolve/"
654 yields the URI "http://xri.example.com/xri-resolve/*home", and the following HTTP request is
655 made to xri.example.com:

```
656    GET /xri-resolve/*home HTTP/1.1
657    If-Modified-Since: Fri, 31 Oct 2003 19:43:32 GMT
658    Accept: application/xrid+xml
659    <other HTTP headers>
```

660 The following HTTP response is received from xri.example.com:

```
661    200 OK HTTP/1.1
662    Content-Type: application/xrid+xml
663    If-Modified-Since: Fri, 31 Oct 2003 19:43:32 GMT
664    <other HTTP headers>
665
666    <XRIDescriptors xmlns="…">
667    <XRIDescriptor>
668    <Resolved>*home</Resolved>
669    <AuthorityID>
670    urn:uuid:925B458F-5907-7654-C3F9-BE3D8912BA73
671    </AuthorityID>
672    <Authority>
673    <AuthorityID>
674    urn:uuid:C9FBEE76-1288-9395-DCD8-DFF35CA9E092
675    </AuthorityID>
676    <URI>
677    http://xri.other.example.com/xri-resolve/*home/
678    </URI>
679    </Authority>
680    <Service>…</Service>
681    …
682    </XRIDescriptor>
683    </XRIDescriptors>
```

Resolving "=example*home*base"

685 Appending the next qualified sub-segment "*base" to the URI "http://xri.other.example.com/xri-
686 resolve/*home/" gives the URI "http://xri.other.example.com/xri-resolve/*home/*base":

```
687     GET /xri-resolve/*home/*base HTTP/1.1
688     If-Modified-Since: Fri, 31 Oct 2003 19:43:32 GMT
689     Accept: application/xrid+xml
690     <other HTTP headers>
```

691 The following HTTP response is received from xri.other.example.com:

```
692     200 OK HTTP/1.1
693     Content-type: application/xrid+xml
694     Expires: Fri, 7 Nov 2003 19:43:33 GMT
695     <other HTTP headers>
696
697     <XRIDescriptors xmlns="…'>
698     <XRIDescriptor>
699     <Resolved>*base</Resolved>
700     <AuthorityID>
701     urn:uuid:C9FBEE76-1288-9395-DCD8-DFF35CA9E092
702     </AuthorityID>
703     <Service>
704     <Type>
705     xri://$res*local.access/X2R
706     </Type>
707     <URI>
708     http://xri.other.example.com/xri-local/base/
709     </URI>
710     <URI>
711     https://xri.other.example.com/xri-local/base/
712     </URI>
713     </Service>
714     …
715     </XRIDescriptor>
716     </XRIDescriptors>
```

717 The result of the final XRI authority resolution step is the set of HTTP and HTTPS URIs shown in
718 the "Service" element above that can be used for local access services (specifically, the X2R
719 local access service as identified by the `xri://$res*local.access/X2R` type).

2.2.5.2 Authority Resolution with Lookahead

721 The next example shows the interaction between a client and server using lookahead resolution
722 for the authority portion of the following XRI:

```
723        xri://=example*home*base/foo*bar
```

724 Assume as in the previous example that the URI for the "=" global context symbol is
725 "http://equals.example.org/xri-resolve". In this example, the client will always request lookahead
726 resolution of all unresolved authority sub-segments.

*Resolving "=example*home*base"*

728 The following HTTP request is made to "equals.example.org":

```
729    GET /xri-resolve/*example*home*base HTTP/1.1
730    If-Modified-Since: Fri, 31 Oct 2003 19:43:31 GMT
731    Accept: application/xrid+xml
732    <other HTTP headers>
```

733 The following HTTP response is received from "equals.example.org" (the content has changed
734 since "Fri, 31 Oct 2003 19:43:31 GMT", the value specified in the "If-Modified-Since" header). The
735 response contains two XRI Descriptor elements, one for "*example" and one for "*home". This
736 indicates to the resolving client that the "equals.example.org" authority has either cached or
737 performed its own resolution to retrieve the descriptor for =example*home:

```
738    200 OK HTTP/1.1
739    Content-Type: application/xrid+xml
740    Expires: Fri, 7 Nov 2003 19:43:31 GMT
741    <other HTTP headers>
742
743    <XRIDescriptors xmlns="…">
744    <XRIDescriptor>
745    <Resolved>*example</Resolved>
746    <AuthorityID>
747    urn:uuid:2BA56CDE-9438-11D9-8BDE-F66BAD1E3F3A
748    </AuthorityID>
749    <Authority>
750    <AuthorityID>
751    urn:uuid:925B458F-5907-7654-C3F9-BE3D8912BA73
752    </AuthorityID>
753    <URI>
754    http://xri.example.com/xri-resolve/
755    </URI>
756    </Authority>
757    <Service>…</Service>
758    </XRIDescriptor>
759    <XRIDescriptor xmlns="…">
760    <Resolved>*home</Resolved>
761    <AuthorityID>
762    urn:uuid:925B458F-5907-7654-C3F9-BE3D8912BA73
763    </AuthorityID>
764    <Authority>
765    <AuthorityID>
766    urn:uuid:C9FBEE76-1288-9395-DCD8-DFF35CA9E092
767    </AuthorityID>
768    <URI>
769    http://xri.other.example.com/xri-resolve/*home/
770    </URI>
771    </Authority>
772    <Service>…</Service>
773    </XRIDescriptor>
774    </XRIDescriptors>
```

775 Note that the XRI Descriptor elements must appear in resolution order, i.e. the first XRI Descriptor
776 describes the authority "*example" and the second describes the authority "*home" within the
777 "*example" namespace.

778 The resolving client, assuming it trusts the resolver's response (see section 3 for more details on
779 trusted resolution), then resolves the "*base" authority sub-segment using the authority URI
780 "http://xri.other.example.com/xri-resolve/*home/" as identified in the last XRI Descriptor above.
781 The following HTTP request is made to "xri.other.example.com":

```
782    GET /xri-resolve/*home/*base HTTP/1.1
783    If-Modified-Since: Fri, 31 Oct 2003 19:43:31 GMT
784    Accept: application/xrid+xml
785    <other HTTP headers>
```

786   The following HTTP response is received from xri.other.example.com:

```
787    200 OK HTTP/1.1
788    Content-type: application/xrid+xml
789    Expires: Fri, 7 Nov 2003 19:43:33 GMT
790    <other HTTP headers>
791
792    <XRIDescriptors xmlns="…">
793    <XRIDescriptor>
794    <Resolved>*base</Resolved>
795    <AuthorityID>
796    urn:uuid:C9FBEE76-1288-9395-DCD8-DFF35CA9E092
797    </AuthorityID>
798    <Service>
799    <Type>
800    xri://$res*local.access/X2R
801    </Type>
802    <URI>
803    http://xri.other.example.com/xri-local/base/
804    </URI>
805    <URI>
806    https://xri.other.example.com/xri-local/base/
807    </URI>
808    </Service>
809    …
810    </XRIDescriptor>
811    </XRIDescriptors>
```

812   Note that the three XRI Descriptor elements in this example (two from the first HTTP resolution
813   from equals.example.org and the one from xri.other.example.com) are exactly the same three
814   XRI Descriptors retrieved from the separate resolution requests showed in section 2.2.5.1.

2.2.6 Resolving Cross-References in XRI Authorities

816   A sub-segment within an XRI authority segment may be a cross-reference. Cross-references are
817   resolved identically to any other sub-segment because the cross-reference is considered opaque
818   by generic XRI resolution. In other words, the value of the cross-reference (including the
819   parentheses) is the literal value of the sub-segment for the purpose of authority resolution.

820   The one exception is a cross-reference rooted on the GCS dollar sign ("$"). The significance of
821   such a cross-reference for resolution depends on the specification that defines the value of the
822   identifier following the $ character. For the XRI suite of specifications, the significance of $ cross-
823   references is defined by the *XRI Metadata Specification* **[XRIMetadata]**. For example, a cross-
824   reference that begins with the GCS dollar sign ("$") followed by the hyphen character ("-"), is
825   specified in **[XRIMetadata]** as insignificant, so this cross-reference and the delimiter that
826   precedes it MUST be ignored entirely during resolution. A cross-reference that begins with the
827   GCS dollar sign ("$") followed by the letter "v", on the other hand, is specified in **[XRIMetadata]**
828   as significant, so this should be treated as a standard cross-reference for the purpose of
829   resolution.

830   Table 5 provides several examples of resolving cross-references. In each example, sub-segment
831   "!b" resolves to an XRI Authority URI of "http://example.com/xri-authority/", and lookahead
832   resolution is not being requested.
833

Cross-reference type	Example XRI	Next Resolution URI after resolving "xri://@!a!b"
Absolute XRI	xri://@!a!b!(@!1!2!3)*e/f	http://example.com/xri-authority/!(@!1!2!3)
Absolute URI	xri://@!a!b*(mailto:jd@example.com)*e/f	http://example.com/xri-authority/*(mailto:jd@example.com)
Relative XRI	xri://@!a!b*(c*d)*e/f	http://example.com/xri-authority/*(c*d)
Metadata XRI (significant)	xri://@!a!b*($v/2.0)*e/f	http://example.com/xri-authority/*($v%2F2.0)
Metadata XRI (ignored)	xri://@!a!b*($-important)*e/f	http://example.com/xri-authority/*e

834 **Table 5: Examples of the Next Authority URIs constructed using different types of cross-references.**

835 2.2.7 XRI Redirects

836 An XRI Descriptor may contain an `xrid:XRIDescriptor/xrid:Synonyms/xrid:External`
837 element and not contain any `xri:XRIDescriptor/xrid:Authority` or
838 `xrid:XRIDescriptor/xrid:LocalAccess` elements. This is called an "XRI redirect"
839 because the XRI Descriptor is effectively redirecting to a new XRI Authority. In this case, the
840 unresolved portion of the original XRI (i.e. the XRI being resolved) is added to the contents of the
841 `/xrid:XRIDescriptor/xrid:Synonyms/xrid:External` element to create a new XRI.
842 This new XRI is then resolved in place of the original XRI.

843 The example in Section 2.2.5 demonstrates the resolution of xri://=example*home*base/foo*bar.
844 The first request is to "equals.example.org". The following XRI redirect could be received as a
845 response.

```
846    200 OK HTTP/1.1
847    Content-Type: application/xrid+xml
848    Expires: Fri, 7 Nov 2003 19:43:31 GMT
849    <other HTTP headers>
850
851    <XRIDescriptors xmlns="...">
852    <XRIDescriptor>
853      <Resolved>*example</Resolved>
854      <AuthorityID>
855        urn:uuid:2BA56CDE-9438-11D9-8BDE-F66BAD1E3F3A
856      </AuthorityID>
857      <Synonyms>
858         <External>
859            xri://@example2
860      </External>
861    </Synonyms>
862    </XRIDescriptor>
863    </XRIDescriptors>
```

864 In this case, a new XRI would be constructed as "xri://=example2*home*base/foo*bar" and the
865 resolution process would begin again with this new XRI.

866 If the original XRI contains additional sub-segments in its Authority component and the
867 `xrid:XRIDescriptor/xrid:Synonyms/xrid:External` element contains a local-path
868 component, the client SHOULD consider this an error condition and fail. For example, consider if

869   the XRI redirect above had been as follows:

870

```
871    200 OK HTTP/1.1
872    Content-Type: application/xrid+xml
873    Expires: Fri, 7 Nov 2003 19:43:31 GMT
874    <other HTTP headers>
875
876    <XRIDescriptors xmlns="...">
877    <XRIDescriptor>
878      <Resolved>*example</Resolved>
879      <AuthorityID>
880        urn:uuid:2BA56CDE-9438-11D9-8BDE-F66BAD1E3F3A
881      </AuthorityID>
882      <Synonyms>
883        <External>
884          xri://@example2/path
885        </External>
886      </Synonyms>
887        …
888    </XRIDescriptor>
889    </XRIDescriptors>
```

890   Now the resulting XRI would be "xri://@example2/path*home*base/foo". Unless the client
891   application has specific reason to believe otherwise, this is an error.

892 2.3 IRI Authority Resolution

893   From the standpoint of generic XRI resolution, an IRI authority segment represents either a DNS
894   name or an IP address at which an XRID for the authority may be retrieved. Requesting the
895   corresponding XRID is a simple matter of making an HTTP(S) GET request using a URI
896   constructed from from the IRI authority segment. The resulting XRI Descriptor is then used to
897   retrieve local access URIs or other XRI authority synonyms or metadata as described in section
898   2.2.

899   The HTTP URI is constructed by extracting the entire IRI authority segment and prepending the
900   string "http://". Then an HTTP GET is performed using an HTTP Accept header containing only
901   the following:

902   `Accept: application/xrid+xml`

903   Additionally, the HTTP GET request MUST have a Host: header (as defined in section 14.23 of
904   **[RFC2616]**) containing the value of the IRI authority segment. The resolving authority MUST use
905   the value of the Host header to populate the `xrid:XRIDescriptor/xrid:Resolved` element
906   in the resulting `xrid:XRIDescriptors` document. For example:

907   `Host: example.com`

908   An HTTP server acting as an IRI authority SHOULD respond with the XRI Descriptors document
909   for that authority.

910   Section 3 of this document defines trusted resolution only for XRI authorities. This document does
911   not define trusted resolution for IRI Authorities. If, however, an IRI authority is known to respond
912   to HTTPS requests (by some means not described in this document) then the resolving client
913   MAY use HTTPS as the access protocol for retrieving the authority's XRID. If the resolving client
914   is satisfied, via transport level security mechanisms, that the response is from the expected IRI
915   authority, then the resolving client may place a higher level of trust on the contents of the XRID
916   than it would have otherwise.

917 The following example demonstrates how the IRI authority segment of the XRI
918 "xri://example.com/local*path" would be resolved into an XRI Descriptor. First the IRI authority is
919 extracted ("example.com"), then the following HTTP Request is made of the server
920 "example.com":

```
921     GET / HTTP/1.1
922     Accept: application/xrid+xml
923     Host: example.com
924     <other HTTP headers>
```

925 The HTTP server acting as the authority might provide the following HTTP response, using the
926 value of the Host header to populate the `xrid:XRIDescriptor/xrid:Resolved` element:

```
927     200 OK HTTP/1.1
928     Content-Type: application/xrid+xml
929     Expires: Fri, 7 Nov 2003 19:43:31 GMT
930     <other HTTP headers>
931
932     <XRIDescriptors xmlns="...">
933     <XRIDescriptor>
934       <Resolved>example.com</Resolved>
935       <AuthorityID>
936         7CF08CE4-9439-11D9-8BDE-F66BAD1E3F3A
937       </AuthorityID>
938       <Synonyms>
939         <External>
940           xri://@example2*path
941         </External>
942        </Synonyms>
943        …
944     </XRIDescriptor>
945     </XRIDescriptors>
```

946 The use of IRI authorities provides backwards compatibility with the large installed base of DNS-
947 and IP-identifiable resources. However, because IRI authorities do not support the additional
948 layer of abstraction and extensibility represented by XRI authority syntax, IRI authorities are not
949 recommended for new deployments of XRI identifiers.

950 2.4 Local Access

951 Local access is the process of interacting with a network endpoint to retrieve a representation of
952 or metadata about a resource identified by an XRI.

953 2.4.1 Local Access Service Types

954 Any number of protocols may be used for local access. This specification defines an HTTP(S)
955 local access protocol with the name "X2R". Other local access services could also be defined—
956 for example, an LDAP or DSML local access protocol that specified the appropriate
957 transformation of the XRI local part into an LDAP distinguished name (including normalization of
958 the XRI local path to the LDAP distinguished name syntax).

959 Work on such additional protocols is left to future specifications. To accommodate such work, this
960 specification reserves a namespace, "$res*local.access", for enumerating local access service
961 types. The "$res" namespace can also be extended by other authorities besides the XRI
962 Technical Committee. See **[XRIMetadata]** for more information about extending "$" namespaces.
963 New local access service types intended for widespread use MUST be identified with XRIs in the
964 $res*local.access namespace. Local access service types defined solely for use within a private
965 or closed community MAY have service types identified by any XRI.

966 2.4.2 The X2R Local Access Service

967 The X2R local access service is derived from the I2R service defined in section 4.3 of
968 **[RFC2483]**. X2R is the default local access service defined in this specification; it is available
969 when the associated `xrid:Descriptor/xrid:Service/xrid:Type` element is not present
970 or when it explicitly contains the value "`xri://$res*local.access/X2R`".

971 X2R is defined as the use of HTTP to interact with a resource using the full extent of the HTTP
972 semantics as defined in **[RFC2616]**. Special attention should be paid to the semantics of the four
973 main HTTP verbs: GET, PUT, POST, and DELETE. For example, clients performing local access
974 typically will use GET to retrieve representations of a resource on the network.

975 This specification does not impose particular semantics beyond what is defined in **[RFC2616]**, but
976 users of this specification are encouraged to review the **[REST]** architecture when building
977 applications using XRIs. Local access is not, however, limited to the REST model of interaction.
978 HTTP local access could be leveraged for the delivery of SOAP messages over HTTP POST, for
979 example, or via use of the GET HTTP verb as a generic read-only operation.

980 The HTTP/HTTPS local access binding defined in this section is flexible enough to be used for a
981 variety of resources. By itself it makes no assumptions about the type of resource identified by
982 the XRI being resolved. However, such metadata can be supplied using the
983 `xrid:XRIDescriptor/xrid:Service/xrid:MediaType` element in an XRID. The resource
984 type may also be established through the context in which the XRI was originally used (e.g., an
985 XML document) or discovered through the HTTP Content-Type header.

986 2.4.2.1 Constructing a Local Access HTTP(S) URI

987 The HTTP(S) URI for X2R local access service is constructed by concatenating the value of any
988 `xrid:XRIDescriptor/xrid:Service/xrid:URI` element in the XRI Descriptor with the
989 URI-normal form of the path portion (matching the "xri-path-absolute" production described in
990 section 2.2.3 of **[XRISyntax]**) of the XRI. If the URI from the XRI Descriptor ends in a forward
991 slash ("/"), this slash MUST be removed before concatenating the path portion.

992 The following pseudocode describes the process for creating, from the local access URI in the
993 XRID, the concrete HTTP(S) URI to which a local access request is made:

```
994    if (local-access-uri ends in "/"):
995       remove trailing "/" in local-access-uri
996
997    local-access-uri = local-access-uri + uri-escape(absolute-path)
```

998 The verb used in the resulting HTTP/(S) request may be any of the verbs defined in **[RFC2616]**,
999 though not all verbs may be supported at every endpoint. All X2R local access endpoints
1000 SHOULD support at least the GET verb, and this should return either a representation of the
1001 identified resource or metadata about the resource.

1002 The full suite of HTTP content negotiation features is available to clients when performing local
1003 access. For example, if the local access service URI is "http://xri.example.com/xri-local", then the
1004 following local access HTTP request for "xri://=example*home/foo*bar" could be made to
1005 "xri.example.com":

```
1006    GET /xri-local/foo*bar HTTP/1.1
1007    If-Modified-Since: Fri, 31 Oct 2003 19:43:33 GMT
1008    <other HTTP headers>
```

1009 The following HTTP response might then be received from xri.example.com:

```
1010        200 OK HTTP/1.1
1011        Expires: Sat, 1 Nov 2003 19:43:33 GMT
1012        Content-Type: text/plain
1013        <other HTTP headers>
1014
1015        This is the result of a local access request.
```

2.5 HTTP Headers

2.5.1 Caching

1018   The HTTP caching capabilities described by **[RFC2616]** should be leveraged for both the default
1019   authority resolution service and the X2R local access service. Specifically, implementations of
1020   XRI resolution SHOULD implement the caching model described in section 13 of **[RFC2616]**. In
1021   particular, the "Expiration Model" of section 13.2 SHOULD be used, as this requires the fewest
1022   round-trip network connections.

1023   All servers providing identifier authority lookup responses SHOULD send the Cache-Control or
1024   Expires headers per section 13.2 of **[RFC2616]** unless there are overriding security or policy
1025   reasons to omit them.

1026   Note that HTTP Cache headers SHOULD NOT conflict with expiration information in an XRID.
1027   That is, the expiration date specified by HTTP caching headers SHOULD NOT be later than any
1028   of the expiration dates for any of the `xrid:XRIDescriptor/xrid:Expires` elements returned
1029   in the HTTP response. This implies that lookahead and proxy resolvers SHOULD compute the
1030   "soonest" expiration date for the XRI Descriptors in a resolution chain and ensure a later date is
1031   not specified by the HTTP caching headers for the HTTP response.

2.5.2 Location

1033   During authority resolution HTTP interaction, "Location" headers may be present per **[RFC2616]**
1034   (i.e., during 3XX redirects). Redirects SHOULD be made cacheable through appropriate HTTP
1035   headers, as specified in section 2.5.1.

2.5.3 Content-Type

1037   For default authority resolution, the "Content-type" header in the 2XX responses MUST contain
1038   the value "`application/xrid+xml`" or "`application/xrid-t-saml+xml`" specifying that
1039   the content is an XRI Descriptor (section 2.2.2) or a trusted XRI Descriptor (section 3.3.1)
1040   respectively.

1041   For X2R local access, clients and servers MAY negotiate content type using standard HTTP
1042   content negotiation features. Regardless of whether this feature is used, however, the server
1043   MUST respond with an appropriate media type in the "Content-type" header if the resource is
1044   found and an appropriate content type is returned.

2.6 Other HTTP Features

1046   HTTP provides a number of other features including transfer-coding, proxying, validation-model
1047   caching, and so forth. All these features may be used insofar as they do not conflict with the
1048   required uses of HTTP described in this document.

2.7 Caching and Efficiency

1050   In addition to HTTP-level caching, resolution clients are encouraged to perform caching at the
1051   application level. For best results, however, resolution clients SHOULD be conservative with
1052   caching expiration semantics, including cache expiration dates. This implies that in a series of

1053 HTTP redirects, for example, the results of the entire process SHOULD only be cached as long
1054 as the shortest period of time allowed by any of the intermediate HTTP responses.

1055 Because not all HTTP client libraries expose caching expiration to applications, identifier
1056 authorities and local access servers SHOULD NOT use cacheable redirects with expiration times
1057 that are sooner than the expiration times of other HTTP responses in the authority resolution
1058 chain or in local access interactions. In general, all XRI deployments should be mindful of
1059 limitations in current HTTP clients and proxies.

1060 For XRI Descriptors, the cache expiration time may also be shortened by the expiration time
1061 provided in the `xrid:XRIDescriptor/xrid:Expires` element (if present). That is, if the
1062 expiration time in `xrid:XRIDescriptor/xrid:Expires` is sooner than the expiration time
1063 calculated from the HTTP caching semantics, then the XRI Descriptor MUST be discarded before
1064 the expiration time in `xrid:XRIDescriptor/xrid:Expires`. Note also that the SAML
1065 assertion used in trusted resolution (section 3) may cause invalidation of a XRI Descriptor even
1066 before HTTP caching semantics or the `xrid:XRIDescriptor/xrid:Expires` element.

1067 With both application-level and HTTP-level caching, the resolution process is designed to have
1068 minimal overhead. In particular, because each qualified sub-segment of an authority identifier is
1069 described by a separate XRI Descriptor, each step of that resolution is independent, and
1070 intermediate results can typically be cached in their entirety. For this reason, resolution of higher-
1071 level (i.e., further to the left) qualified sub-segments, which are common to more identifiers, will
1072 naturally result in a greater number of cache hits than resolution of lower-level sub-segments.

3 Trusted Resolution

3.1 Introduction

This section defines a method for performing trusted XRI authority resolution as an extension of the generic XRI resolution protocol defined in section 2 of this document.

This trusted resolution protocol does not provide a means to encrypt the contents of resolution requests and responses, nor does it provide a means for a responder to provide different responses for different requestors. These services may be provided by other security protocols used in conjunction with this specification, but confidentiality and client-authentication are explicitly out of scope in this version of this specification.

This section assumes the reader is familiar with, at a minimum, the ABNF defined in Appendix A of **[XRISyntax]** and the generic resolution protocol defined in section 2 of this document.

3.2 Overview and Example (Non-Normative)

Trusted XRI Authority resolution is a straightforward enhancement to generic XRI resolution. The client application requests resolution of one or more qualified sub-segments in the XRI Authority segment exactly as described in section 2 of this document with one exception: instead of using "`application/xrid+xml`" in the "Accept" header of the HTTP(S) request, a content type of "`application/xrid-t-saml+xml`" is used. The XRI Authority responds with an XRI Descriptor that contains an additional element - a digitally signed SAML **[SAML]** assertion that asserts the validity of the containing XRI Descriptor. If the response does not contain a valid, digitally signed SAML assertion (as defined in section 3.2 of this document), this is considered an error condition, and trusted resolution MUST NOT proceed.

The following example steps through resolution of the authority portion of the same XRI used in Section 2 of this document:

```
xri://=example*home*base/foo.bar
```

As in standard resolution, there is no defined discovery process for the trusted resolution URI(s) of the community root – it must be known *a priori* and is expected to be part of the configuration of the resolver. A recommended practice is to publish an XRI Descriptor containing a valid SAML assertion signed by the community root. In this example, assume the `xrid:Authority/xrid:URI` element of the XRI Descriptor for the global community root "=" specifies that the URI for the "=" global context symbol is "http://equals.example.org/xri-resolve".

In trusted resolution, each XRI Authority is associated with an identifier called an AuthorityID. An AuthorityID is a URI, or an XRI in URI-normal form, uniquely associated with a particular XRI Authority. Each XRI Authority MUST have at least one AuthorityID, and no two XRI Authorities can have the same AuthorityID. The AuthorityID of the community root, like the community root's URI, is defined in the `xrid:XRIDescriptor/xrid:Authority/xrid:AuthorityID` element of the community root's XRI Descriptor (or its equivalent). For this example, assume the AuthorityID for the "=" global context symbol is "`urn:uuid:498FB006-B9EF-4943-B10A-A71FC2ED1B89`". For more information on `xrid:XRIDescriptor/xrid:AuthorityID`, see Section 3.3.3 below.

Finally, in trusted resolution, each XRI Authority is associated with some key used to verify digital signatures. The key for the community root must be known and configured in advance. If an XRI Descriptor is used to describe the community root, information about this key may be found in the `xrid:XRIDescriptor/xrid:Authority/ds:KeyInfo` element of that document.

1116 Note that the digital signatures in the following examples are for reference only. The digest values
1117 are not valid and the signatures will not verify.

1118 ### *Resolving "=example"*

1119 The following HTTP request is made to "equals.example.org":

```
1120     GET /xri-resolve/*example HTTP/1.1
1121     If-Modified-Since: Fri, 31 Oct 2003 19:43:31 GMT
1122     Accept: application/xrid-t-saml+xml
1123     <other HTTP headers>
```

1124 **Example 1: Request for =example**

1125 This request contains an Accept header with the value "`application/xrid-t-saml+xml`".
1126 The client is requesting a response that contains a signed SAML assertion. If the resolving client
1127 will accept either trusted or generic resolution, preferring trusted resolution, it could have used the
1128 value "`application/xrid-t-saml+xml, application/xrid+xml`" for the Accept header.

1129 The following HTTP response is received from "equals.example.org":

```
1130    200 OK HTTP/1.1
1131    Content-Type: application/xrid-t-saml+xml
1132    Expires: Fri, 7 Nov 2003 19:43:31 GMT
1133    <other HTTP headers>
1134
1135    <XRIDescriptors
1136      xmlns="xri://$res*schema/XRIDescriptor">
1137    <XRIDescriptor
1138      xrid:id="baec221f3c0f17f53ca6839989632056">
1139     <Resolved>*example</Resolved>
1140     <AuthorityID>urn:uuid:498FB006-B9EF-4943-B10A-A71FC2ED1B89
1141     </AuthorityID>
1142     <Authority>
1143      <AuthorityID>urn:uuid:C5C9EFDF-A3BC-4301-88C6-B1AE0AD6DA77
1144      </AuthorityID>
1145      <URI xrid:trusted="true">http://xri.example.com/xri-resolve/</URI>
1146      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
1147      …
1148      </ds:KeyInfo>
1149     </Authority>
1150     <TrustMechanism>xri://$res*trusted/XRITrusted</TrustMechanism>
1151     <saml:Assertion
1152       Version="2.0"
1153       ID="_ad9571ad-cd23-85e2-e928-abba20b6c424"
1154       IssueInstant="2004-07-01T00:46:02Z"
1155       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
1156      <saml:Issuer>xri://@example</saml:Issuer>
1157      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
1158       <ds:SignedInfo>
1159         <ds:CanonicalizationMethod
1160           Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1161         <ds:SignatureMethod
1162           Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
1163         <ds:Reference URI="#baec221f3c0f17f53ca6839989632056">
1164         <ds:Transforms>
1165          <ds:Transform
1166            Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
1167    signature" />
1168          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
1169    c14n#">
1170           <ec:InclusiveNamespaces
1171             xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
1172             PrefixList="#default code ds kind rw saml samlp typens" />
1173         </ds:Transform>
1174        </ds:Transforms>
1175        <ds:DigestMethod
1176          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1177        <ds:DigestValue>BSsnowZG5DYV0X0C8GAeB1cvLzw=</ds:DigestValue>
1178       </ds:Reference>
1179      </ds:SignedInfo>
1180      <ds:SignatureValue>
1181    kE9p35G4mcombsqEztJMX1R3J26gwc4cbjSz5fUv3aVg3j/iLhrbf0qKywYNMLdQMjBRcCg
1182    5N1l0
1183    Kvv2UrgvQ5kgQ9dm7/563rRzKAaIQwMopZpTFli4eXw+nc8XEH+KnXdu/R9DHOg9k0BKIF6
1184    BGk07
1185    xC6Q9X+byQWenPjAZ1c=
1186      </ds:SignatureValue>
1187     </ds:Signature>
1188     <saml:Subject>
1189      <saml:NameID NameQualifier="urn:uuid:498FB006-B9EF-4943-B10A-
1190    A71FC2ED1B89">
1191    *example
1192      </saml:NameID>
```

```
1193          </saml:Subject>
1194          <saml:Conditions
1195            NotBefore="2004-06-01T00:00:00Z"
1196            NotOnOrAfter="2004-09-01T00:00:00Z" />
1197          <saml:AttributeStatement>
1198           <saml:Attribute Name="xri://$res*schema/XRIDescriptor">
1199
1200      <saml:AttributeValue>#baec221f3c0f17f53ca6839989632056</saml:AttributeV
1201      alue>
1202            </saml:Attribute>
1203          </saml:AttributeStatement>
1204        </saml:Assertion>
1205      </XRIDescriptor>
1206      </XRIDescriptors>
```

1207   **Example 2 – Response for =example**

1208   The response contains an `xrid:XRIDescriptor/saml:Assertion` element that includes an
1209   assertion about the validity of the XRI Descriptor. (For more information about SAML assertions
1210   in XRI Descriptors, see section 3.3.3.) The response also contains an
1211   `xrid:XRIDescriptor/xrid:Authority/ds:KeyInfo` element. This required element
1212   informs the client that XRI Descriptors digitally signed by the described XRI Authority are to be
1213   verified using this key.

1214   Finally, note that two instances of `xrid:AuthorityID` appear in the XRI Descriptor: one as a
1215   child of `xrid:XRIDescriptor` and the other as a child of `xrid:Authority`. The child of
1216   `xrid:XRIDescriptor` is the AuthorityID of the *current describing* authority (the one publishing
1217   this XRI Descriptor) and matches the expected AuthorityID of the community root
1218   (urn:uuid:498FB006-B9EF-4943-B10A-A71FC2ED1B89). The child of the `xrid:Authority`
1219   element contains the AuthorityID of the *described* XRI Authority (the authority being described
1220   within the `xrid:Authority` element). Responses from that XRI Authority will contain this
1221   AuthorityID as a child of `xrid:XRIDescriptor`.

1222   The client validates the signed SAML assertion as described in Section 3.3 before continuing.

1223   ### *Resolving "=example*home"*

1224   Appending the next qualified sub-segment "*home" to the URI "http://xri.example.com/xri-resolve/"
1225   yields the URI "http://xri.example.com/xri-resolve/*home". The following HTTP request with an
1226   Accept header value of "`application/xrid-t-saml+xml`" is made to "xri.example.com":

```
1227      GET /xri-resolve/*home HTTP/1.1
1228      Accept: application/xrid-t-saml+xml
1229      <other HTTP headers>
```

1230   **Example 3 – Request for *home**

1231   The following HTTP response is received from xri.example.com:

```
1232    200 OK HTTP/1.1
1233    Content-Type: application/xrid-t-saml+xml
1234    If-Modified-Since: Fri, 31 Oct 2003 19:43:32 GMT
1235    <other HTTP headers>
1236
1237    <XRIDescriptors
1238      xmlns="xri://$res*schema/XRIDescriptor">
1239    <XRIDescriptor
1240    xrid:id="1f81b6e0-b64b-1026-f1bc-c0a80b9d3f5b">
1241     <Resolved>*home</Resolved>
1242     <AuthorityID>urn:uuid:C5C9EFDF-A3BC-4301-88C6-B1AE0AD6DA77
1243     </AuthorityID>
1244     <Authority>
1245      <AuthorityID>urn:uuid:A9F28515-AB03-4883-8852-8EECB54CE1D5
1246      </AuthorityID>
1247    <URI xrid:trusted="true">
1248       http://xri.example.com/xri-resolve/*home/
1249    </URI>
1250      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
1251      …
1252      </ds:KeyInfo>
1253     </Authority>
1254     <Service>…</Service> <!-- Local Access Service -->
1255     <TrustMechanism>xri://$res*trusted/XRITrusted</TrustMechanism>
1256     <saml:Assertion
1257       Version="2.0"
1258       ID="_66f1f3e0-b64b-1026-34a4-c0a80b9d59c1"
1259       IssueInstant="2004-05-01T00:46:03Z"
1260       xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
1261      <saml:Issuer>xri://@example</saml:Issuer>
1262      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
1263       <ds:SignedInfo>
1264         <ds:CanonicalizationMethod
1265           Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
1266         <ds:SignatureMethod
1267           Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
1268         <ds:Reference URI="#1f81b6e0-b64b-1026-f1bc-c0a80b9d3f5b">
1269         <ds:Transforms>
1270          <ds:Transform
1271            Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
1272    signature" />
1273          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
1274    c14n#">
1275            <ec:InclusiveNamespaces
1276              xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
1277              PrefixList="#default code ds kind rw saml samlp typens" />
1278          </ds:Transform>
1279         </ds:Transforms>
1280         <ds:DigestMethod
1281           Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
1282         <ds:DigestValue>BSsnowZG5DYV0X0C8GAeB1cvLzw=</ds:DigestValue>
1283        </ds:Reference>
1284       </ds:SignedInfo>
1285       <ds:SignatureValue>
1286    kE9p35G4mcombsqEztJMX1R3J26gwc4cbjSz5fUv3aVg3j/iLhrbf0qKywYNMLdQMjBRcCg
1287    5N1l0
1288    Kvv2UrgvQ5kgQ9dm7/563rRzKAaIQwMopZpTFli4eXw+nc8XEH+KnXdu/R9DHOg9k0BKIF6
1289    BGk07
1290    xC6Q9X+byQWenPjAZ1c=
1291       </ds:SignatureValue>
1292      </ds:Signature>
1293      <saml:Subject>
```

```
1294        <saml:NameID NameQualifier="urn:uuid:C5C9EFDF-A3BC-4301-88C6-
1295    B1AE0AD6DA77">
1296    *home
1297        </saml:NameID>
1298      </saml:Subject>
1299      <saml:Conditions
1300        NotBefore="2004-06-01T00:00:00Z"
1301        NotOnOrAfter="2004-09-01T00:00:00Z" />
1302      <saml:AttributeStatement>
1303       <saml:Attribute Name="xri://$res*schema/XRIDescriptor">
1304        <saml:AttributeValue>#1f81b6e0-b64b-1026-f1bc-c0a80b9d3f5b
1305    </saml:AttributeValue>
1306       </saml:Attribute>
1307      </saml:AttributeStatement>
1308     </saml:Assertion>
1309    </XRIDescriptor>
1310    </XRIDescriptors>
```

1311   **Example 4 – Response for *home**

1312   The client validates the SAML assertion as described in Section 3.3 before continuing.

1313   ### *Resolving "=example*home*base"*

1314   Appending the next qualified sub-segment "*base" to the URI
1315   "http://xri.example.com/xri-resolve/*home/" gives the URI
1316   "http://xri.example.com/xri-resolve/*home/*base". This is the target of the next trusted resolution
1317   request, again with the Accept header value "application/xrid-t-saml+xml":

```
1318    GET /xri-resolve/*home/*base HTTP/1.1
1319    If-Modified-Since: Fri, 31 Oct 2003 19:43:32 GMT
1320    Accept: application/xrid-t-saml+xml
1321    <other HTTP headers>
```

1322   **Example 5 – Request for *base**

1323   The following HTTP response is received from xri.example.com:

```
200 OK HTTP/1.1
Content-type: application/xrid-t-saml+xml
Expires: Fri, 7 Nov 2003 19:43:33 GMT
<other HTTP headers>

<XRIDescriptors
  xmlns="xri://$res*schema/XRIDescriptor">
<XRIDescriptor
  xrid:id="7600e1a0-b64d-1026-ea89-c0a80b9d3814">
  <Resolved>*base</Resolved>
  <AuthorityID>urn:uuid:A9F28515-AB03-4883-8852-8EECB54CE1D5
  </AuthorityID>
  <Service>
    <Type>xri://$res*local.access/X2R</Type>
    <URI>http://xri.example.com/xri-local/base/</URI>
    <URI>https://xri.example.com/xri-local/base/</URI>
  </Service>
  <TrustMechanism>xri://$res*trusted/XRITrusted</TrustMechanism>
 <saml:Assertion
   Version="2.0"
   ID="_1a6a12d0-b64d-1026-c1ba-c0a80b9db964"
   IssueInstant="2004-06-03T00:46:03Z"
   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
  <saml:Issuer>xri://@example</saml:Issuer>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
   <ds:SignedInfo>
     <ds:CanonicalizationMethod
       Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
     <ds:SignatureMethod
       Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
     <ds:Reference URI="#7600e1a0-b64d-1026-ea89-c0a80b9d3814">
     <ds:Transforms>
      <ds:Transform
        Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature" />
       <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#">
         <ec:InclusiveNamespaces
           xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
           PrefixList="#default code ds kind rw saml samlp typens" />
      </ds:Transform>
     </ds:Transforms>
     <ds:DigestMethod
       Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
     <ds:DigestValue>BSsnowZG5DYV0X0C8GAeB1cvLzw=</ds:DigestValue>
    </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue>
kE9p35G4mcombsqEztJMX1R3J26gwc4cbjSz5fUv3aVg3j/iLhrbf0qKywYNMLdQMjBRcCg
5N1l0
Kvv2UrgvQ5kgQ9dm7/563rRzKAaIQwMopZpTFli4eXw+nc8XEH+KnXdu/R9DHOg9k0BKIF6
BGk07
xC6Q9X+byQWenPjAZ1c=
   </ds:SignatureValue>
  </ds:Signature>
  <saml:Subject>
   <saml:NameID NameQualifier="urn:uuid:A9F28515-AB03-4883-8852-
8EECB54CE1D5">
*example
   </saml:NameID>
  </saml:Subject>
  <saml:Conditions
    NotBefore="2004-06-03T00:46:03Z"
```

```
1387        NotOnOrAfter="2004-12-01T00:00:00Z" />
1388      <saml:AttributeStatement>
1389       <saml:Attribute Name="xri://$res*schema/XRIDescriptor">
1390        <saml:AttributeValue>#7600e1a0-b64d-1026-ea89-
1391   c0a80b9d3814</saml:AttributeValue>
1392       </saml:Attribute>
1393      </saml:AttributeStatement>
1394     </saml:Assertion>
1395     …
1396    </XRIDescriptor>
1397    </XRIDescriptors>
```

1398   **Example 6 – Response for *base**

1399   The SAML assertion is validated as described in Section 3.3 before proceeding. The result of the
1400   final XRI Authority resolution step is the set of HTTP and HTTPS URIs shown in the
1401   `xrid:XRIDescriptor/xrid:Service` element above that can be used for local access
1402   services (in this case, X2R service).

3.3 Trusted Resolution Protocol

1404   This section normatively defines client and server behavior in trusted resolution.

3.3.1 XML Elements and Attributes

1406   Three elements of an XRI Descriptor defined in section 2.2.2 have limited usage in generic
1407   resolution but play a critical role in trusted resolution.

xrid:XRIDescriptor/xrid:AuthorityID

1409   Always required, but critical in trusted resolution for identification of the current describing
1410   authority.

xrid:XRIDescriptor/xrid:Authority/xrid:AuthorityID

1412   Always required, but critical for trusted resolution for identification of the target described
1413   authority.

xrid:XRIDescriptor/xrid:TrustMechanism

1415   Required when providing trusted resolution. A URI or XRI in URI-normal form that
1416   specifies the mechanism used to provide trusted resolution. The URI for the trust
1417   mechanism defined in this specification is "xri://$res*trusted/XRITrusted".

1418   In addition, one element from the SAML **[SAML]** namespace is also critical for verifying the
1419   results of trusted resolution.

xrid:XRIDescriptor/saml:Assertion

1421   Required when providing trusted resolution. A SAML assertion from the *describing*
1422   Authority (the one providing the XRI Descriptor) that asserts that the describing authority
1423   believes the information contained in the enclosing XRI Descriptor is correct. Because
1424   the assertion is digitally signed and the digital signature encompasses the containing XRI
1425   Descriptor, it also provides a mechanism for the recipient to detect unauthorized changes
1426   since the time the XRI Descriptor was published.

1427   Note that while a `saml:Issuer` element is required within a `saml:Assertion` element,
1428   this specification makes no requirement as to the value of the `saml:Issuer` element. It
1429   is up to the community root to place restrictions, if any, on the `saml:Issuer` element. A
1430   suitable approach is to use an XRI in URI-Normal Form that describes the organization
1431   providing responses for the XRI Authority (e.g. xri://@example).

1432 Finally, trusted resolution adds several new elements and attributes to XRI Descriptors to assist
1433 in verifying XRIDs produced by the described authority (i.e., the next authority in the resolution
1434 chain that is being described by the `xrid:XRIDescriptor/xrid:Authority` element of the
1435 current XRID).

xrid:XRIDescriptor/xrid:Authority/xrid:URI/@trusted

1437   Optional. Default value of "false" (or "0"). Indicates whether this service endpoint is
1438   capable of returning trusted resolution results. If the value is "1" or "true", the *described*
1439   authority is willing to return signed XRI Descriptors at this URI.

xrid:XRIDescriptor/xrid:Authority/ds:KeyInfo

1441   Required when providing trusted resolution. Provides the key data needed to validate an
1442   XRI Descriptor provided by the *described* Authority as a result of resolution at the
1443   described Authority. This element comprises the key distribution method for trusted XRI
1444   resolution.

1445 Figure 5 below demonstrates the relationship between these elements for two descriptors in a
1446 resolution chain: one describing an authority, and one produced by the authority being described.
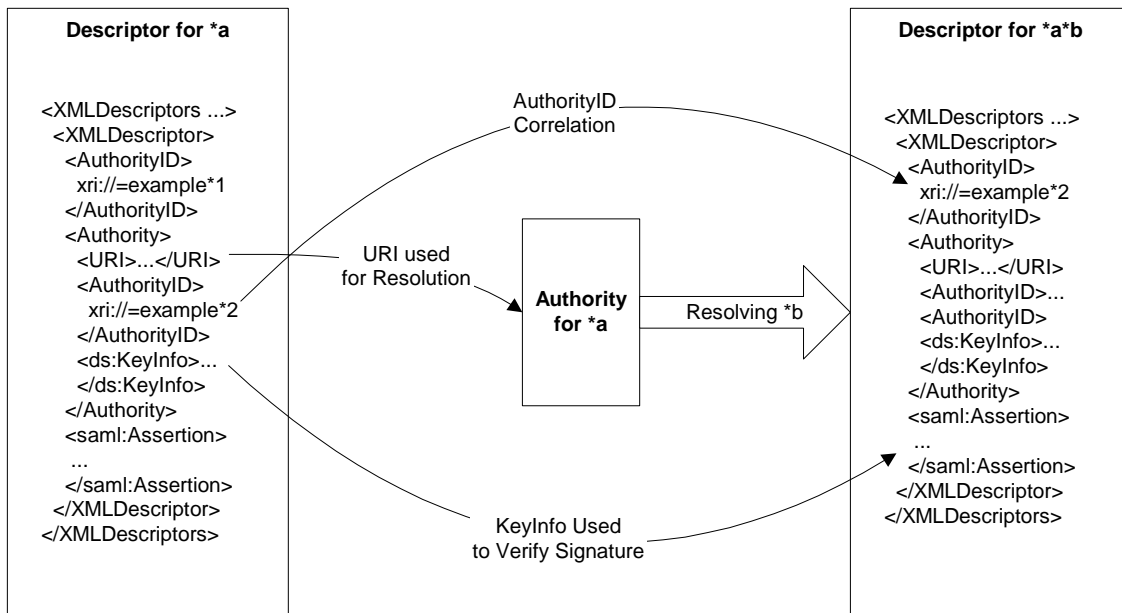
1447
1448 **Figure 5: Correlation of XRID Elements for Trusted Resolution**

3.3.2 Use and Correlation of AuthorityID Elements

1450 Each XRI Authority participating in trusted resolution MUST be associated with at least one
1451 AuthorityID, and this AuthorityID MUST NOT ever be assigned to any other XRI Authority. In
1452 other words, AuthorityID is a persistent unique identifier for a particular XRI Authority.

1453 An AuthorityID may be any valid URI that meets the requirements of permanence and
1454 uniqueness described above. Examples of appropriate URIs include URNs as defined by
1455 **[RFC2141]** and fully persistent XRIs converted to URI-Normal Form as defined by **[XRISyntax]**.

1456 Conceptually, AuthorityID assures a resolving client that the returned XRI Descriptor has not
1457 been maliciously replaced with a similar XRI Descriptor from a second, and possibly
1458 unauthorized, XRI Authority. To prevent this type of attack, the XRI Descriptor must be explicitly
1459 associated with a specific unique XRI Authority, and the client must have some means of
1460 verifying this association. The `xrid:XRIDescriptor/xrid:AuthorityID` element provides
1461 this explicit association.

1462 There is no defined discovery process for the AuthorityID of the community root; it must be
1463 published in the root XRID (or other equivalent description document) and verified independently.
1464 The AuthorityID for an XRI Authority other than the community root is furnished by the
1465 `xrid:XRIDescriptor/xrid:Authority/xrid:AuthorityID` element in the XRI
1466 Descriptor that describes the authority.

3.3.3 Client Behavior

1468 From a client's perspective, trusted resolution is identical to the generic resolution protocol
1469 described in section 2 of this document with the addition of the following REQUIRED behavior:

1470 • The client MUST indicate to the resolving server that a signed XRI Descriptor is desired.
1471 This is accomplished by adding an HTTP Accept header with the media type identifier
1472 "`application/xrid-t-saml+xml`". Clients willing to accept either trusted or untrusted
1473 resolution descriptors may use a combination of "`application/xrid-t-saml+xml`" and
1474 "`application/xrid+xml`" in the Accept header as described in section 14.1 of
1475 **[RFC2616]**. Media type identifiers SHOULD be ordered according to the client's preference
1476 for the media type of the response.

1477 • The client SHOULD NOT request trusted resolution from an authority unless the
1478 corresponding `xrid:Descriptor/xrid:Authority/xrid:URI` element has a "trusted"
1479 attribute with the value of "true" or "1".

1480 • Each XRI Descriptor in a resolution chain MUST be individually validated using the rules
1481 described in this section. When `xrid:XRIDescriptor` elements may come both from
1482 freshly-retrieved XRID documents and from a local cache, an implementation MUST ensure
1483 that these requirements are satisfied each time a resolution request is performed.

1484 The client MUST confirm that each `xrid:XRIDescriptor` element contains a
1485 `saml:Assertion` element as an immediate child, and that this assertion is valid per the
1486 processing rules described by **[SAML]**. In addition, the following requirements MUST be met:

1487 • The `saml:Assertion` must contain a valid enveloped digital signature as defined by
1488 **[XMLDSig]** and constrained by Section 5.4 of **[SAML]**.

1489 • The signature must apply to the `xrid:XRIDescriptor` element that contains the
1490 signed SAML assertion. Specifically, the signature must contain a single
1491 `ds:SignedInfo/ds:Reference` element, and the `URI` attribute of this reference must
1492 refer to the id (`xrid:id` attribute) of the `xrid:XRIDescriptor` element that is the
1493 immediate parent of the signed SAML assertion.

1494 • If the digital signature enveloped by the SAML assertion contains a `ds:KeyInfo`
1495 element, the client MAY reject the signature if this key does not match the signer's expected
1496 key, as specified by the `ds:KeyInfo` element present in the XRI Descriptor that was used to
1497 describe the current authority. For example, if Authority A provides an XRI Descriptor
1498 describing Authority B, and this XRID has an
1499 `xrid:XRIDescriptor/xrid:Authority/ds:KeyInfo` element that describes the key
1500 used to validate descriptors produced by Authority B, this key is Authority B's "expected key"
1501 and should be used when validating XRI Descriptor elements produced by Authority B. For a
1502 community root authority, the expected key is known *a priori* as part of the configuration in
1503 the client for that particular community root.

1504 • The client confirms that the value of the `xrid:XRIDescriptor/xrid:Resolved`
1505 element matches the sub-segment whose resolution resulted in the current XRI Descriptor.

1506 • The client confirms that the value of the `xrid:XRIDescriptor/xrid:AuthorityID`
1507 element matches the XRI Authority's "expected AuthorityID". As with the key information, the
1508 "expected AuthorityID" is the value of
1509 `xrid:XRIDescriptor/xrid:Authority/xrid:AuthorityID` in the XRI Descriptor
1510 that describes the current Authority. As before, for a community root authority, the XRI

1511 Authority's expected AuthorityID is known *a priori* and is part of the configuration in the client
1512 for that particular community root.

1513 • The client confirms that the value of the `xrid:XRIDescriptor/xrid:AuthorityID`
1514 element matches the value of the `NameQualifier` attribute of the
1515 `xrid:XRIDescriptor/saml:Assertion/saml:Subject/saml:NameID` element.

1516 • The client confirms that the value of the xrid:XRIDescriptor/xrid:Resolved element
1517 matches the value of the xrid:XRIDescriptor/saml:Assertion/saml:Subject/saml:NameID
1518 element.

1519 • The client confirms that the value of the
1520 `xrid:XRIDescriptor/xrid:TrustMechanism` is "xri://$res*trusted/XRITrusted".

1521 • The client confirms the existence of exactly one
1522 `xrid:XRIDescriptor/saml:Assertion/saml:AttributeStatment` with exactly one
1523 `saml:Attribute` element that has a `Name` attribute of "xri://$res*schema/XRIDescriptor".
1524 This `saml:Attribute` element must have exactly one `saml:AttributeValue` element
1525 whose text value is a URI reference to the `xrid:id` attribute of the `xrid:XRIDescriptor`
1526 element that is the immediate parent of the signed SAML assertion.

1527 If any of the above requirements are not met for an XRI Descriptor in the resolution chain, the
1528 result MUST NOT be considered a valid trusted resolution response as defined by this document.
1529 Note that this does not preclude a client from considering alternative resolution paths. For
1530 example, if two URIs are listed under an `xrid:Authority` element and the response from one
1531 fails to meet the requirements above, the client may repeat the validation process using the
1532 second URI. If the second URI passes the tests, it may be considered a trusted resolution
1533 response as defined by this document and trusted resolution may continue.

1534 3.3.4 Server Behavior

1535 From the server's perspective, trusted resolution is identical to the generic resolution protocol
1536 described in section 2 of this document with the addition of the following behavior. This behavior
1537 is REQUIRED if a resolution client requests trusted resolution as described in section 3.2 and the
1538 server intends to honor the client's request.

1539 If, during the HTTP(S) request/response interaction, the server agrees to return a trusted
1540 resolution response (indicated by the content type of "`application/xrid-t-saml+xml`"), the
1541 XRI Descriptor returned by the server must contain a `saml:Assertion` element as an
1542 immediate child of `xrid:XRIDescriptor` that is valid per the processing rules described by
1543 **[SAML]**. In addition, the following requirements MUST be met:

1544 • The SAML Assertion MUST contain a valid enveloped digital signature as defined by
1545 **[XMLDSig]** and as constrained by section 5.4 of **[SAML]**.

1546 • The signature MUST apply to the `xrid:XRIDescriptor` element that contains the
1547 signed SAML assertion. Specifically, the signature must contain a single
1548 `ds:SignedInfo/ds:Reference` element, and the `URI` attribute of this reference MUST
1549 refer to the `xrid:XRIDescriptor` element that is the immediate parent of the signed SAML
1550 assertion. The URI reference MUST NOT be empty; it MUST refer to the identifier contained
1551 in the `xrid:id` attribute of the `xrid:XRIDescriptor` element.

1552 • The digital signature enveloped by the SAML assertion is allowed to contain a
1553 `ds:KeyInfo` element. If it is included, it MUST describe the key used to verify the digital
1554 signature element. Because the signing key is known in advance by the resolution client, the
1555 `ds:KeyInfo` element SHOULD be omitted from the digital signature. Because the client is
1556 required to verify the digital signature using the key obtained from the `xrid:Authority`
1557 element describing the current authority, it is important that the server sign such that the
1558 signature can be verified using the `ds:KeyInfo` element registered in the XRI Descriptor(s)
1559 that describes this authority.

1560 • The `xrid:Resolved` element MUST be present, and the value of this field MUST match
1561   the XRI Authority sub-segment requested by the client.

1562 • The `xrid:XRIDescriptor` element MUST have an `xrid:AuthorityID` element as
1563   an immediate child. The value of the `xrid:AuthorityID` element MUST be the Authority
1564   ID, as described in Section 3.2, of the responding XRI Authority.

1565 • The `xrid:XRIDescriptor/xrid:TrustMechanism` MUST be present and the value
1566   MUST be "xri://$res*trusted/XRITrusted".

1567 • The `xrid:XRIDescriptor/saml:Subject/saml:NameID` element MUST be
1568   present and equal to the `xrid:XRIDescriptor/xrid:Resolved element`.

1569 • The `NameQualifier` attribute of the
1570   `xrid:XRIDescriptor/saml:Assertion/saml:Subject/saml:NameID` element
1571   MUST be present and equal to the `xrid:XRIDescriptor/xrid:AuthorityID` element.

1572 • There MUST be exactly one `saml:AttributeStatement` present in the
1573   `xrid:XRIDescriptor/saml:Assertion element`. It MUST contain exactly one
1574   `saml:Attribute` element with a `Name` attribute of "xri://$res*schema/XRIDescriptor". This
1575   `saml:Attribute` element MUST contain exactly one `saml:AttributeValue` element
1576   whose text value is a URI reference to the `xrid:id` attribute of the `xrid:XRIDescriptor`
1577   that is an immediate parent of the `saml:Assertion` element.

1578 If a resolving client requests trusted resolution *and* lookahead resolution, the responding authority
1579 SHOULD attempt to perform trusted resolution on behalf of the client as described in section 3.
1580 However, the server providing lookahead resolution MUST NOT return untrusted XRIDs if the
1581 client requests trusted resolution. If the server cannot obtain trusted XRIDs for the additional
1582 lookahead sub-segments, it SHOULD return only the trusted XRIDs it has obtained and allow the
1583 client to continue.

3.3.5 Additional Requirements of Authorities Offering Trusted Resolution

1586 The `xrid:XRIDescriptor/xrid:Authority` element that describes an authority
1587 participating in trusted resolution as defined by this specification ("the described XRI Authority")
1588 has the following requirements:

1589 • The `trusted` attribute of the `xrid:XRIDescriptor/xrid:Authority/xrid:URI`
1590   element MUST contain the value "1" or "true".

1591 • The `xrid:XRIDescriptor/xrid:Authority` element MUST contain a `ds:KeyInfo`
1592   element as an immediate child. The value of this element MUST be the key that validates
1593   digital signatures created by the described XRI Authority.

1594 • The `xrid:XRIDescriptor/xrid:Authority` element MUST contain an
1595   `xrid:AuthorityID` element as an immediate child. The value of this field MUST be the
1596   AuthorityID of the described XRI Authority, i.e. the value that will appear in the
1597   `xrid:Descriptor/xrid:AuthorityID` element of an XRI Descriptor returned from the
1598   described XRI Authority.

1599 • In addition, an identifier community SHOULD publish an XRI Descriptor for the
1600   community root that meets the requirements listed above and it SHOULD make that XRI
1601   Descriptor easily available to relevant parties.

4 Extensibility and Versioning

4.1 Extensibility

XRI Descriptors use an an open-content schema because they are designed to be extended with other metadata. In a number of places, extension elements and attributes from namespaces other than "xri://$res*schema/XRIDescriptor*($v%2F2.0)" are explicitly allowed. These extension points are designed to simplify default processing of XRI Descriptors using a "Must Ignore" rule. The base rule is that unrecognized elements and attributes, and the content and child elements of unrecognized elements, MUST be ignored. As a consequence, elements that would normally be recognized by a processor MUST be ignored if they appear as descendants of an unrecognized element.

Extension elements MUST NOT require new interpretation of elements defined in this document. That is, if an extension element is present, a processor must be able to ignore it and still correctly process the Descriptor document.

Extension specifications MAY simulate "Must Understand" behavior by applying an "enclosure" pattern. Elements defined by the XRI Descriptor schema whose meaning or interpretation are to be modified by extension elements can be wrapped in a extension container element that is defined by the extension specification. This extension container element SHOULD be in the same namespace as the extension elements that must be understood by the consumer of the XRI Descriptor. All elements whose interpretations are modified by the extension will now be contained in an element (the extension container element) that will be ignored by consumers unable to process the extension.

The following example illustrates this pattern using an extension container element from an extension namespace ("other:SuperAuthority") that contains an extension element ("other:ExtensionElement"):

```
<XRIDescriptor>
  <other:SuperAuthority>
    <Authority>
       …
       <other:ExtensionElement>…</other:ExtensionElement>
    </Authority>
  </other:SuperAuthority>
  <Service>
     …
  </Service>
</XRIDescriptors>
```

In this example, the other:ExtensionElement modifies the interpretation or processing rules for the parent xrid:Authority element and therefore must be understood by the consumer for the proper interpretation of the parent xrid:Authority element. To preserve the correct interpretation of the xrid:Authority element in this context, the xrid:Authority element is "wrapped" so only consumers that understand elements in the other:SuperAuthority namespace will attempt to process the xrid:Authority element.

4.1.1 Specific Points of Extensibility

The use of HTTP and XML in the design of the generic resolution service, the trusted resolution service, and the X2R local access service provide the following specific points of extensibility:

- Specification of new authority resolution service types (xrid:Authority/xrid:Type in the XRI Descriptor).

1648 • Specification of new local access service types (`xrid:Service/xrid:Type` in the XRI
1649 Descriptor).

1650 • Specification of new trust mechanisms (`xrid:TrustMechanism` in the XRI Descriptor).
1651 For example, an existing secure private network in which resolution is intrinsically trustworthy
1652 may wish to express its own trust mechanism explicitly.

1653 • HTTP negotiation of content types, language, encoding, etc.

1654 • Use of HTTP verbs such as POST, PUT and DELETE during local access.

1655 • Use of HTTP redirects (3XX) or other response codes defined by **[RFC2616]** during
1656 identifier authority resolution or X2R local access.

1657 • Use of cross-references within XRIs, particularly for associating new types of metadata
1658 with a resource.

1659 4.2 Versioning

1660 Versioning of the XRI specification set is expected to be occur infrequently. Experience, however,
1661 demonstrates that such versioning is eventually inevitable. For this reason, this section describes
1662 versioning guidelines.

1663 When version information is expressed as both a Major and Minor version, it is expressed in the
1664 form *Major.Minor*. The version number $Major_B.Minor_B$ is higher than the version number
1665 $Major_A.Minor_A$ if and only if:

1666 $Major_B > Major_A$ OR (($Major_B = Major_A$) AND $Minor_B > Minor_A$)

1667 4.2.1 Versioning of the XRI Resolution Specification

1668 New releases of the XRI Resolution specification may specify changes to the resolution protocol
1669 and/or to resolution data structures. When changes affect either of these, the resolution
1670 specification version number will be changed. Where changes are purely editorial, the version
1671 number will not be changed.

1672 In general, if a change is backward-compatible, the new version will be identified using the
1673 current major version number and a new minor version number. If the change is not backward-
1674 compatible, the new version will be identified with a new major version number.

1675 4.2.2 Versioning of XRI Descriptor Elements

1676 Both the `xrid:XRIDescriptors` element and the `xrid:XRIDescriptor` element have
1677 Version attributes. The value of these attributes MUST be the version value of the specification to
1678 which their containing elements conform.

1679 When new versions of the XRI Resolution specification are released, the namespace for the XRI
1680 Descriptor schema may or may not be changed. If there is a major version number change, the
1681 namespace for the `xrid:XRIDescriptors` document is likely to change. If there is only a minor
1682 version number change, the namespace for the `xrid:XRIDescriptors` document may remain
1683 unchanged.

1684 In general, maintaining namespace stability and adding to or changing the content of a schema
1685 are competing goals. While certain design strategies can facilitate such changes, it is difficult to
1686 predict how existing implementations will react to any given change, making forward compatibility
1687 difficult to achieve. Nevertheless, the right to make such changes in minor revisions is reserved.
1688 Except in special circumstances (for example, to correct major deficiencies or to fix errors),
1689 implementations should expect forward-compatible schema changes in minor revisions, allowing
1690 new messages to validate against older schemas.

1691 Implementations SHOULD expect, and be prepared to deal with, new extensions and message
1692 types in accordance with the processing rules laid out for those types. Minor revisions may

1693 introduce new types that leverage the extension facilities described in Section 4.1. Older
1694 implementations SHOULD reject such extensions gracefully when they are encountered in
1695 contexts with specific semantic requirements.

4.2.3 Versioning of Protocols

1697 Both the authority resolution and local access protocols defined in this document may also be
1698 versioned by future releases of the XRI Resolution specification. If these protocols are not
1699 backward-compatible with older implementations, they will likely get a new XRI for use in
1700 identifying them in XRI Descriptors.

1701 Note that it is possible for version negotiation to happen in the protocol itself. For example, HTTP
1702 provides a mechanism to negotiate the version of the HTTP protocol being used. If and when an
1703 authority resolution or local access protocol provides its own version-negotiation mechanism, the
1704 specification is likely to continue to use the same XRI to identify the protocol as was used in
1705 previous versions of the XRI Resolution specification.

1706 5 Security and Data Protection

1707 Significant portions of this specification deal directly with security issues, and these will not be
1708 summarized again here. In addition, basic security practices and typical risks in resolution
1709 protocols are well-documented in many other specifications. Only security considerations directly
1710 relevant to XRI resolution are included here.

1711 5.1 DNS Spoofing

1712 As the specified resolution mechanism is dependent on DNS, the accuracy of the XRI resolution
1713 response is dependent on the accuracy of the original DNS query. When trustable, unambiguous
1714 and authoritative responses are required, trusted resolution as defined by this specification is
1715 recommended. With trusted resolution as defined by this specification, resolution results can be
1716 evaluated independently of DNS resolution results. While this does not solve the problem of DNS
1717 spoofing, it does allow the client to detect an error condition and reject the resolution result as
1718 untrustworthy. For environments that require higher confidence in the result of DNS resolution,
1719 DNSSEC **[DNSSEC]** is recommended as a supplement to trusted resolution as defined by this
1720 specification.

1721 5.2 HTTP Security

1722 Many of the security considerations set forth in HTTP/1.1 **[RFC2616]** apply to XRI Resolution
1723 protocols defined here. In particular, confidentiality of the communication channel is not
1724 guaranteed by HTTP. Server-authenticated HTTPS should be considered in cases where
1725 confidentiality of resolution requests and responses is desired.

1726 Special consideration should be given to proxy and caching behaviors to ensure accurate and
1727 reliable responses from resolution requests. For various reasons, network topologies increasingly
1728 have transparent proxies, some of which may insert VIA and other headers as a consequence, or
1729 may even cache content without regard to caching policies set by a resource's HTTP authority.

1730 Implementations of XRI Proxies and caching authorities should also take special note of the
1731 security recommendations in HTTP/1.1 **[RFC2616]** section 15.7

1732 5.3 Caching Authorities

1733 In addition to traditional HTTP caching proxies, XRI resolution authority proxies may be a part of
1734 the resolution topology. Such proxies should take special precautions against cache poisoning,
1735 as these caching entities may represent trust decision points within a deployment's resolution
1736 architecture.

1737 5.4 Lookahead and Proxy Resolution

1738 During proxy resolution, some or all of the XRI Authority is provided to the proxy resolver. During
1739 lookahead resolution, sub-segments of the XRI Authority for which the resolving network endpoint
1740 is not authoritative may be revealed to that endpoint.

1741 In both cases, privacy considerations should be evaluated before disclosing such information.

1742 5.5 SAML Considerations

1743 Trusted resolution must adhere to the rules defined by the SAML 2.0 Core Specification.
1744 Particularly noteworthy are the XML Transform restrictions on XML Signature defined in SAML
1745 and the enforcement of the SAML Conditions element regarding the validity period.

5.6 Community Root Authorities

The XRI Authority information for a community root needs to be well-known to the clients that request resolution within that community. For trusted resolution, this includes the URIs, the `AuthorityID`, and the `ds:KeyInfo` information. An acceptable means of providing this information is for the community root authority to produce a self-signed XRI Descriptor and publish it to a server-authenticated HTTPS endpoint. Special care should be taken to ensure the correctness of such an XRID; if this information is incorrect, an attacker may be able to convince a client of an incorrect result during trusted resolution.

5.7 Denial-Of-Service Attacks

XRI Resolution, including trusted resolution, is vulnerable to denial-of-service (DOS) attacks typical of systems relying on DNS and HTTP.

5.8 Limitations of Trusted Resolution

While the trusted resolution mechanism specified in this document provides a way to verify the integrity of a successful XRI resolution, it does not provide a way to verify the integrity of a resolution failure. Reasons for this limitation include the prevalence of non-malicious network failures, the existence of denial-of-service attacks, and the ability of a man-in-the-middle attacker to modify HTTP responses when resolution is not performed over HTTPS.

Additionally, there is no revocation mechanism for the keys used in trusted resolution. Therefore, a signed resolution's validity period should be limited appropriately to mitigate the risk of an incorrect or invalid resolution.

6 References

6.1 Normative

[DNSSEC]   D. Eastlake, *Domain Name System Security Extensions*, http://www.ietf.org/rfc/rfc2535, RFC 2535, March 1999.

[RFC2046]   N. Borenstein, N. Freed, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, http://www.ietf.org/rfc/rfc2046.txt, RFC 2046, November 1996.

[RFC2119]   S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, RFC 2119, March 1997.

[RFC2141]   R. Moats, *URN Syntax*, http://www.ietf.org/rfc/rfc2141.txt, IETF RFC 2141, May 1997.

[RFC2483]   M. Meallling, R. Daniel Jr., *URI Resolution Services Necessary for URN Resolution,* http://www.ietf.org/rfc/rfc2483.txt, RFC 2483, January 1999.

[RFC2616]   R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, *Hypertext Transfer Protoc– -- HTTP/1.1*, http://www.ietf.org/rfc/rfc2616.txt, RFC 2616, June 1999.

[SAML]   S. Cantor, J. Kemp, R. Philpott, E. Maler, *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0,* http://www.oasis-open.org/committees/security, March 2005.

[XMLDSig]   D. Eastlake, J. Reagle, D. Solo et al., *XML-Signature Syntax and Processing*, World Wide Web Consortium, http://www.w3.org/TR/xmldsig-core/, February 12, 2002.

[XRIMetadata]   D. Reed, *Extensible Resource Identifier (XRI) Metadata V2.0*, http://docs.oasis-open.org/xri/xri/V2.0/xri-metadata-V2.0-cd-01.pdf, March 2005.

[XRISyntax]   D. Reed, D. McAlpin, *Extensible Resource Identifier (XRI) Syntax V2.0*, http://docs.oasis-open.org/xri/xri/V2.0/xri-syntax-V2.0-cd-01.pdf, March 2005.

6.2 Informative

[REST]   http://internet.conveyor.com/RESTwiki/moin.cgi/FrontPage

[XRIIntro]   D. Reed, D. McAlpin, *Introduction to XRIs*, http://docs.oasis-open.org/xri/xri/V2.0/xri-intro-V2.0.pdf, Work-In-Progress, March 2005.

[XRIReqs]   G. Wachob, D. Reed, M. Le Maitre, D. McAlpin, D. McPherson, *Extensible Resource Identifier (XRI) Requirements and Glossary v1.0*, http://www.oasis-open.org/apps/org/workgroup/xri/download.php/2523/xri-requirements-and-glossary-v1.0.doc, June 2003.

Appendix A. XML Schema for XRI Descriptor (Normative)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="xri://$res*schema/XRIDescriptor*($v%2F2.0)"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xrid="xri://$res*schema/XRIDescriptor*($v%2F2.0)" elementFormDefault="qualified">
        <!-- Utility patterns -->
        <xs:attributeGroup name="otherattribute">
                <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:attributeGroup>
        <xs:group name="otherelement">
                <xs:choice>
                        <xs:any namespace="##other" processContents="lax"/>
                        <xs:any namespace="##local" processContents="lax"/>
                </xs:choice>
        </xs:group>
        <xs:complexType name="URIpattern">
                <xs:simpleContent>
                        <xs:extension base="xs:anyURI">
                                <xs:attributeGroup ref="xrid:otherattribute"/>
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>
        <xs:complexType name="Stringpattern">
                <xs:simpleContent>
                        <xs:extension base="xs:string">
                                <xs:attributeGroup ref="xrid:otherattribute"/>
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>
        <!-- Patterns for elements -->
        <xs:element name="XRIDescriptors">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="xrid:XRIDescriptor"
maxOccurs="unbounded"/>
                                <xs:group ref="xrid:otherelement" minOccurs="0"
maxOccurs="unbounded"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="xrid:otherattribute"/>
                        <xs:attribute ref="xrid:version"/>
                </xs:complexType>
        </xs:element>
        <xs:element name="XRIDescriptor">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="xrid:Resolved" />
                                <xs:element ref="xrid:AuthorityID" />
                                <xs:element ref="xrid:Expires" minOccurs="0"/>
                                <xs:element ref="xrid:Authority" minOccurs="0"
maxOccurs="unbounded"/>
                                <xs:element ref="xrid:Service" minOccurs="0"
maxOccurs="unbounded"/>
                                <xs:element ref="xrid:Synonyms" minOccurs="0"/>
                                <xs:element ref="xrid:TrustMechanism" minOccurs="0"/>
                                <xs:group ref="xrid:otherelement" minOccurs="0"
maxOccurs="unbounded"/>
                        </xs:sequence>
                        <xs:attribute ref="xrid:id"/>
                        <xs:attributeGroup ref="xrid:otherattribute"/>
                        <xs:attribute ref="xrid:version"/>
                </xs:complexType>
        </xs:element>
        <xs:element name="Resolved" type="xrid:Stringpattern"/>
        <xs:element name="Expires">
```

```
1868                    <xs:complexType>
1869                        <xs:simpleContent>
1870                            <xs:extension base="xs:dateTime">
1871                                <xs:attributeGroup ref="xrid:otherattribute"/>
1872                            </xs:extension>
1873                        </xs:simpleContent>
1874                    </xs:complexType>
1875            </xs:element>
1876            <xs:element name="Authority">
1877                <xs:complexType>
1878                    <xs:sequence>
1879                        <xs:element ref="xrid:AuthorityID" minOccurs="0"/>
1880                        <xs:element ref="xrid:Type" minOccurs="0"/>
1881                        <xs:group ref="xrid:TrustableURI" maxOccurs="unbounded"/>
1882                        <xs:group ref="xrid:otherelement" minOccurs="0"
1883    maxOccurs="unbounded"/>
1884                    </xs:sequence>
1885                    <xs:attributeGroup ref="xrid:otherattribute"/>
1886                </xs:complexType>
1887            </xs:element>
1888            <xs:element name="AuthorityID" type="xrid:URIpattern"/>
1889            <xs:element name="Type" type="xrid:URIpattern"/>
1890            <xs:group name="TrustableURI">
1891                <xs:sequence>
1892                    <xs:element name="URI">
1893                        <xs:complexType>
1894                            <xs:simpleContent>
1895                                <xs:extension base="xrid:URIpattern">
1896                                    <xs:attribute ref="xrid:trusted"/>
1897                                </xs:extension>
1898                            </xs:simpleContent>
1899                        </xs:complexType>
1900                    </xs:element>
1901                </xs:sequence>
1902            </xs:group>
1903            <xs:element name="Service">
1904                <xs:complexType>
1905                    <xs:sequence>
1906                        <xs:element ref="xrid:Type" minOccurs="0"/>
1907                        <xs:group ref="xrid:URI" maxOccurs="unbounded"/>
1908                        <xs:element ref="xrid:MediaType" minOccurs="0"
1909    maxOccurs="unbounded"/>
1910                        <xs:group ref="xrid:otherelement" minOccurs="0"
1911    maxOccurs="unbounded"/>
1912                    </xs:sequence>
1913                    <xs:attributeGroup ref="xrid:otherattribute"/>
1914                </xs:complexType>
1915            </xs:element>
1916            <xs:group name="URI">
1917                <xs:sequence>
1918                    <xs:element name="URI" type="xrid:URIpattern"/>
1919                </xs:sequence>
1920            </xs:group>
1921            <xs:element name="MediaType" type="xrid:Stringpattern"/>
1922            <xs:element name="Synonyms">
1923                <xs:complexType>
1924                    <xs:sequence>
1925                        <xs:choice minOccurs="0" maxOccurs="unbounded">
1926                            <xs:element ref="xrid:Internal"/>
1927                            <xs:element ref="xrid:External"/>
1928                        </xs:choice>
1929                        <xs:group ref="xrid:otherelement" minOccurs="0"
1930    maxOccurs="unbounded"/>
1931                    </xs:sequence>
1932                    <xs:attributeGroup ref="xrid:otherattribute"/>
1933                </xs:complexType>
1934            </xs:element>
1935            <xs:element name="Internal" type="xrid:URIpattern"/>
1936            <xs:element name="External" type="xrid:URIpattern"/>
1937            <xs:element name="TrustMechanism" type="xrid:URIpattern"/>
1938            <xs:attribute name="version" type="xs:string" fixed="2.0"/>
```

```
1939        <xs:attribute name="trusted" type="xs:boolean"/>
1940        <xs:attribute name="id" type="xs:ID"/>
1941    </xs:schema>
```

Appendix B. RelaxNG Compact Syntax Schema for XRI Descriptor (Non-normative)

```
1944   namespace xrid="xri://$res*schema/XRIDescriptor*($v%2F2.0)"
1945   namespace xml="http://www.w3.org/XML/1998/namespace"
1946   namespace local=""
1947
1948
1949   start=XRIDescriptors
1950
1951   # Utility patterns
1952   anything = ( element * {anything} | attribute * {text} | text ) *
1953   otherattribute = attribute *-(xrid:*|local:*) {text}
1954   otherelement = element *-xrid:* {anything}
1955   URIpattern = (xsd:anyURI, otherattribute *)
1956   Stringpattern = (xsd:string, otherattribute *)
1957   versionattribute = attribute xrid:version {text}
1958   idattribute = attribute xrid:id {xsd:ID}
1959
1960   ###########################
1961   # XRIDescriptors Container
1962   XRIDescriptors = element xrid:XRIDescriptors {
1963       versionattribute,
1964       XRIDescriptor+,
1965       XRIDescriptors-ex-elem,
1966       XRIDescriptors-ex-attr
1967   }
1968
1969   # XRIDescriptors Extension
1970   XRIDescriptors-ex-elem = otherelement *
1971   XRIDescriptors-ex-attr = otherattribute *
1972
1973   ###########################
1974   # XRIDescriptor Definition
1975   XRIDescriptor = element xrid:XRIDescriptor {
1976       attribute xrid:id {xsd:ID}?,
1977       versionattribute,
1978       Resolved,
1979       AuthorityID,
1980       Expires ?,
1981       Authority *,
1982       Service *,
1983       Synonyms ?,
1984       TrustMechanism ?,
1985       XRIDescriptor-ex-elem,
1986       XRIDescriptor-ex-attr
1987   }
1988
1989   # XRIDescriptor Extension
1990   XRIDescriptor-ex-elem = otherelement *
1991   XRIDescriptor-ex-attr = otherattribute *
1992
1993   #####################
1994   # Resolved Definition
1995   Resolved = element xrid:Resolved { Resolved-content}
1996
1997   # Resolved Extension
1998   Resolved-content = Stringpattern
1999
2000   ####################
2001   # Expires Definition
2002   Expires = element xrid:Expires {
2003       xsd:dateTime,
2004       Expires-ex-attr
2005   }
2006
```

```
2007     # Expires Extension
2008     Expires-ex-attr = otherattribute *
2009
2010     ######################
2011     # Authority Definition
2012     Authority = element xrid:Authority {
2013         AuthorityID?,
2014         Type?,
2015         TrustableURI+,
2016         Authority-ex-attr,
2017         Authority-ex-elem
2018     }
2019
2020     # Authority Extension
2021     Authority-ex-attr = otherattribute *
2022     Authority-ex-elem = otherelement *
2023
2024     ########################
2025     # AuthorityID Definition
2026     AuthorityID = element xrid:AuthorityID { AuthorityID-content}
2027
2028     # AuthorityID extension
2029     AuthorityID-content = URIpattern
2030
2031     #################
2032     # Type Definition
2033     Type = element xrid:Type { Type-content}
2034
2035     # Type Extension
2036     Type-content = URIpattern
2037
2038     ################
2039     # Trustable URI Definition
2040     TrustableURI = element xrid:URI { TrustableURI-content }
2041
2042     TrustableURI-content = (
2043         URIpattern,
2044         attribute xrid:trusted {xsd:boolean}?
2045     )
2046
2047     ###################
2048     # Service Definition
2049     Service = element xrid:Service {
2050         Type?,
2051         URI+,
2052         MediaType *,
2053         Service-ex-attr,
2054         Service-ex-elem
2055     }
2056     # Service Extension
2057     Service-ex-attr = otherattribute *
2058     Service-ex-elem = otherelement *
2059
2060     ######################################
2061     # URI Definition (for Service element)
2062     URI = element xrid:URI { URI-content }
2063
2064     # URI Extension
2065     URI-content = URIpattern
2066
2067     #####################
2068     # MediaType Definition
2069     MediaType = element xrid:MediaType { MediaType-content }
2070
2071     # MediaType Extension
2072     MediaType-content = URIpattern
2073
2074     ####################
2075     # Synonyms Definition
2076     Synonyms = element xrid:Synonyms {
2077         (
```

```
2078            Internal &
2079            External
2080        )+,
2081        Synonyms-ex-attr,
2082        Synonyms-ex-elem
2083    }
2084
2085    Synonyms-ex-attr = otherattribute *
2086    Synonyms-ex-elem = otherelement *
2087
2088    #####################
2089    # Internal Definition
2090    Internal = element xrid:Internal { Internal-content }
2091
2092    # Internal Extension
2093    Internal-content = URIpattern
2094
2095    #####################
2096    # External Definition
2097    External = element xrid:External { External-content }
2098
2099    # External Extension
2100    External-content = URIpattern
2101
2102    ##########################
2103    # TrustMechanism Definition
2104
2105    TrustMechanism = element xrid:TrustMechanism { TrustMechanism-content }
2106
2107    # TrustMechanism Extension
2108    TrustMechanism-content = URIpattern
```

Appendix C. Acknowledgments

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

- Geoffrey Strongin, Advanced Micro Devices
- Ajay Madhok, AmSoft Systems
- Jean-Jacques Dubray, Attachmate
- William Barnhill, Booz Allen and Hamilton
- Drummond Reed, Cordance Corporation
- Marc Le Maitre, Cordance Corporation
- Dave McAlpin, Epok
- Loren West, Epok
- Peter Davis, NeuStar
- Masaki Nishitani, Nomura Research
- Nat Sakimura, Nomura Research
- Tetsu Watanabe, Nomura Research
- Owen Davis, PlaNetwork
- Victor Grey, PlaNetwork
- Fen Labalme, PlaNetwork
- Mike Lindelsee, Visa International
- Gabriel Wachob, Visa International
- Dave Wentker, Visa International
- Bill Washburn, XDI.ORG

The editors also would like to acknowledge the following people for their contributions to previous versions of the OASIS XRI specifications (affiliations listed for OASIS members):

Thomas Bikeev, EAN International; Krishna Sankar, Cisco; Winston Bumpus, Dell; Joseph Moeller, EDS; Steve Green, Epok; Lance Hood, Epok; Adarbad Master, Epok; Davis McPherson, Epok; Chetan Sabnis, Epok; Phillipe LeBlanc, GemPlus; Jim Schreckengast, Gemplus; Xavier Serret, Gemplus; John McGarvey, IBM; Reva Modi, Infosys; Krishnan Rajagopalan, Novell; Tomonori Seki, NRI; James Bryce Clark, OASIS; Marc Stephenson, TSO; Mike Mealling, Verisign; Rajeev Maria, Visa International; Terence Spielman, Visa International; John Veizades, Visa International; Lark Allen, Wave Systems; Michael Willett, Wave Systems; Matthew Dovey; Eamonn Neylon; Mary Nishikawa; Lars Marius Garshol; Norman Paskin; Bernard Vatant.

- A special acknowledgement to Jerry Kindall (Epok) for a full editorial review.

2142 Appendix D. Notices

2143 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
2144 that might be claimed to pertain to the implementation or use of the technology described in this
2145 document or the extent to which any license under such rights might or might not be available;
2146 neither does it represent that it has made any effort to identify any such rights.

2147 Information on OASIS's procedures with respect to rights in OASIS specifications can be found at
2148 the OASIS website. Copies of claims of rights made available for publication and any assurances
2149 of licenses to be made available, or the result of an attempt made to obtain a general license or
2150 permission for the use of such proprietary rights by implementors or users of this specification,
2151 can be obtained from the OASIS President.

2152 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
2153 applications, or other proprietary rights which may cover technology that may be required to
2154 implement this specification. Please address the information to the OASIS President.

2155 **Copyright © OASIS Open 2005.** *All Rights Reserved.*

2156 This document and translations of it may be copied and furnished to others, and derivative works
2157 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
2158 published and distributed, in whole or in part, without restriction of any kind, provided that the
2159 above copyright notice and this paragraph are included on all such copies and derivative works.
2160 However, this document itself does not be modified in any way, such as by removing the
2161 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
2162 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
2163 Property Rights document must be followed, or as required to translate it into languages other
2164 than English.

2165 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
2166 successors or assigns.

2167 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2168 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
2169 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
2170 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
2171 PARTICULAR PURPOSE.