



Extensible Resource Identifier (XRI) Generic Syntax and Resolution Specification

Committee Draft, 12 January 2004

Document identifier:

xri-syntax-resolution-1.0-cd

Location:

<http://www.oasis-open.org/committees/xri/xri-syntax-resolution-1.0-cd>

Editors:

Gabe Wachob, Visa International <gwachob@visa.com>
Drummond Reed, OneName <drummond.reed@onename.com>
Dave McAlpin, Epok <dave.mcalpin@epok.net>
Mike Lindelsee, Visa International <mlindels@visa.com>
Peter Davis, Neustar <peter.davis@neustar.biz>
Nat Sakimura, NRI <n-sakimura@nri.co.jp>

Abstract:

This document is the normative technical specification for XRI generic syntax and resolution. For a non-normative introduction to the uses and features of XRIs, see the "XRI Primer" at <http://www.oasis-open.org/committees/xri/xri-primer-1.0>.

Status:

This document is a *Committee Draft*.

Committee members should send comments on this specification to the xri@lists.oasis-open.org list. Others should subscribe to and send comments to the xri-comment@lists.oasis-open.org list. To subscribe, send an email message to xri-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the XRI TC web page (<http://www.oasis-open.org/committees/xri/>).

The errata page for this specification is at <http://www.oasis-open.org/committees/xri/xri-syntax-resolution-1.0-errata>.



34 Table of Contents

35	Introduction	55
36	1.1 Overview of XRIs	55
37	1.1.1 Generic Syntax	55
38	1.1.2 Examples	66
39	1.1.3 URI, URL, URN, and XRI	77
40	1.2 Design Considerations	77
41	1.2.1 Abstraction and Independence	77
42	1.2.2 Persistence and Reassignability	88
43	1.2.3 Human-Friendliness and Machine-Friendliness	88
44	1.2.4 Internationalization	88
45	1.2.5 Cross-Context Identification	88
46	1.2.6 Authority, Delegation, and Federation	88
47	1.2.7 Security and Privacy	88
48	1.2.8 Extensibility	88
49	1.3 Terminology and Notation	88
50	1.3.1 Keywords	88
51	1.3.2 Syntax Notation	99
52	1.3.3 Glossary	99
53	2 Syntax	1313
54	2.1 Syntax Components	1313
55	2.1.1 Authority	1313
56	2.1.1.1 URI Authority	1313
57	2.1.1.2 XRI Authority	1545
58	2.1.1.3 Global Context Symbols (GCS)	1545
59	2.1.1.4 Cross-References	1646
60	2.1.1.5 Self-References	1646
61	2.1.2 Path	1646
62	2.1.3 Query	1747
63	2.1.4 Fragment	1747
64	2.2 Characters	1818
65	2.2.1 Character Encoding	1818
66	2.2.2 Reserved Characters	1818
67	2.2.3 Unreserved Characters	1818
68	2.2.4 Escaped Characters	1949
69	2.2.4.1 Escaped Encoding	1949
70	2.2.4.2 Encoding XRI Metadata	1949
71	2.2.4.3 Transforming XRIs into IRIs and URIs	2020
72	2.2.4.4 Special Escaping Rules for XRI Syntax	2124
73	2.2.4.5 Transforming URIs and IRIs Back into XRIs	2222
74	2.2.5 Excluded Characters	2323
75	2.3 Relative XRI References	2424
76	2.3.1 Establishing a Base XRI	2424

77	2.3.2 Obtaining the Referenced XRI.....	2424
78	2.3.3 Leading Segments Containing a Colon.....	2525
79	2.4 Normalization and Comparison.....	2525
80	2.4.1 Case.....	2626
81	2.4.2 Encoding, Escaping, and Transformations.....	2626
82	2.4.3 Optional Syntax.....	2626
83	2.4.4 Cross-References.....	2626
84	2.4.5 Canonicalization.....	2727
85	3 Resolution.....	2828
86	3.1 Introduction.....	2828
87	3.1.1 Assumptions.....	2828
88	3.1.2 Phases of Resolution.....	2828
89	3.1.3 URI vs. XRI Authorities.....	2929
90	3.1.4 XRI Metadata Reserved for XRI Resolution.....	2929
91	3.2 XRI Authority Resolution.....	2929
92	3.2.1 Overview.....	2929
93	3.2.2 XRI Descriptors.....	3030
94	3.2.3 Initiating Resolution.....	3232
95	3.2.4 Iterating Resolution.....	3232
96	3.2.5 Examples.....	3333
97	3.2.6 Resolving Cross-References in XRI Authorities.....	3535
98	3.2.7 User Relative XRIs.....	3636
99	3.3 URI Authority Resolution.....	3636
100	3.4 Local Access.....	3636
101	3.4.1 Local Access Service Types.....	3636
102	3.4.2 HTTP/HTTPS Local Access.....	3636
103	3.4.3 Constructing a Local Access HTTP/HTTPS URI.....	3737
104	3.4.4 Using a Cross-Reference to Specify a Representation Type.....	3838
105	3.5 HTTP Headers.....	3838
106	3.5.1 Caching.....	3838
107	3.5.2 Location.....	3838
108	3.5.3 Content-Location.....	3939
109	3.5.4 Content-Type.....	3939
110	3.5.5 X-XRI-Canonical.....	3939
111	3.6 Other HTTP Features.....	3939
112	3.7 Caching and Efficiency.....	3939
113	3.8 Points of Extensibility.....	4040
114	4 Security and Data Protection.....	4144
115	4.1 Secure Resolution.....	4144
116	4.2 XRI Metadata.....	4144
117	4.3 XRI Usage in Legacy Infrastructure.....	4144
118	4.4 XRI Usage in Evolving Infrastructure.....	4144
119	5 References.....	4242
120	5.1 Normative.....	4242

121	5.2 Informative.....	4343
122	Appendix A. Collected ABNF for XRI (Normative)	4444
123	Appendix B. XML Schema for XRI Descriptor (Normative).....	4747
124	Appendix C. Transforming HTTP URIs to XRIs (Non-Normative).....	4949
125	Appendix D. Acknowledgments.....	5050
126	Appendix E. Revision History	5151
127	Appendix F. Notices	5252
128		

129 Introduction

130 1.1 Overview of XRIs

131 An Extensible Resource Identifier (XRI) provides a standard means of abstractly identifying a
132 resource independent of any particular concrete representation of that resource—or, in the case
133 of a completely abstract resource, of any representation at all.

134 XRIs are similar to URIs as defined in “*Uniform Resource Identifiers (URI): Generic Syntax*”
135 [RFC2396], but contain additional syntactic elements and extend the unreserved character set to
136 include characters beyond those allowed in generic URIs. To accommodate applications that
137 expect generic URIs, this specification defines rules for transforming an XRI into a valid URI as
138 defined by [RFC2396]. Since a revision of RFC 2396 is currently in progress, the XRI scheme
139 also incorporates some simplifications and enhancements to generic URI syntax as proposed in
140 [RFC2396bis].

141 XRI syntax is internationalized following the recommendations in “*Guidelines for New URL*
142 *Schemes*” [RFC2718] and “*Extensible Markup Language (XML) 1.0 (Second Edition)*” [XML], and
143 specifically the requirements of the “anyURI” datatype as specified in “XML Schema Part 2:
144 Datatypes” [XMLSchema2]. To do this, the XRI scheme incorporates the syntax recommended
145 in another work-in-progress, “*Internationalized Resource Identifiers (IRIs)*” [IRI].

146 Although an XRI is not a Uniform Resource Name (URN) as defined in “*URN Syntax*” [RFC2141],
147 XRIs consisting entirely of persistent segments are designed to meet the requirements set out in
148 “*Functional Requirements for Uniform Resource Names*” [RFC1737].

149 This document specifies the ABNF for the XRI scheme. In addition it specifies an HTTP-based
150 resolution protocol for XRIs. Use of this protocol is not required; XRIs may also be resolved using
151 other protocols or resolution mechanisms.

152 While [RFC2396bis] and [IRI] are cited in this document, they are both works in progress and are
153 consequently non-normative. All relevant information from these proposals is reproduced here, so
154 access to these documents, while very informative, is not required.

155 1.1.1 Generic Syntax

156 XRI syntax is designed to be as simple and extensible as URI syntax. A fully-qualified XRI
157 consists of the scheme name “xri:” followed by the same four optional components as a generic
158 URI.

159

```
160 xri: authority / path ? query # fragment
```

161

162 The definitions of these components are, for the most part, supersets of the equivalent
163 components in the generic URI syntax. One advantage of this approach is that the vast majority
164 of HTTP URIs, which inherit directly from generic URI syntax, can be transformed to valid XRIs
165 simply by changing the scheme from “http” to “xri”. The rules for this transformation are
166 summarized in Appendix C, “Transforming HTTP URIs to XRIs”.

167 XRI syntax extends generic URI syntax in six ways by providing support for:

- 168 1. *Persistent and reassignable segments*. Generic URI syntax does not distinguish between
169 persistent and reassignable identifiers. XRI syntax enables the top-level authority
170 segment as well as any subsequent path segment to be explicitly designated as either
171 persistent or reassignable.

- 172 2. *Unlimited delegation*. Generic URI syntax supports delegated identifiers (i.e., DNS names
173 or IP addresses) only within the top-level authority segment. XRI syntax supports
174 delegation of both persistent and reassignable identifiers at any level of the path.
- 175 3. *Global context symbols*. While XRI syntax supports the same generic syntax used in
176 URIs for DNS and IP authorities, it also provides shorthand symbols for establishing the
177 abstract context of an identifier.
- 178 4. *Cross-references*. Generic URI syntax does not provide a way to share identifiers across
179 contexts. This capability is particularly useful with abstract identifiers (e.g., to establish
180 the generic type of a resource, or to share standardized identifier metadata such as
181 version indicators). For this reason, XRI syntax allows XRIs (and URIs) to be shared
182 across contexts by means of parenthetical nesting.
- 183 5. *Self-references*. Generic URI syntax does not provide a way to indicate whether or not a
184 URI is intended for resolution. Since an XRI may itself be the full representation of a
185 abstract non-network resource (for example, concepts like “love,” “honesty,” or “user-
186 friendliness”), XRI syntax provides a way to express self-reference.
- 187 6. *Internationalized character set*. Generic URI syntax limits legal characters to a subset of
188 the US-ASCII character set. XRI syntax, following the lead of Internationalized Resource
189 Identifiers [IRI], employs the broader Unicode character set, making the use of XRIs in
190 languages other than English much more straightforward.

191 1.1.2 Examples

192 The following examples illustrate XRI syntax. These examples have minimal annotation and are
193 only intended to give a sense of the scope and flavor of XRI syntax. For more information on the
194 normative syntax, see section 2. For a complete description of the uses and features of XRIs, see
195 the non-normative XRI Primer.
196

```
197 xri://www.example.com/pages/index.html  
198     --standard HTTP URI converted to an XRI  
199  
200 xri://[2010:836B:4179::836B:4179]/pages/index.html  
201     --using an IPv6 authority per RFC 2732  
202  
203 xri://www.example.com/inventory.parts/widget.subwidget.foobarator  
204     --delegation of reassignable identifiers  
205  
206 xri://www.example.com/:inventory:parts/:12:7:234  
207     --delegation of persistent identifiers  
208  
209 xri:@ExampleCorp  
210 xri:@ExampleCorp.www  
211 xri:@ExampleCorp.website  
212 xri:=JohnDoe  
213 xri:=JohnDoe.home  
214 xri:=JohnDoe.work  
215 xri:+flowers  
216 xri:+flowers.rose  
217 xri:+flowers.daisy  
218     --global context symbols  
219  
220 xri://www.example.com/(+management)/(+CEO)  
221 xri:(urn:oasis:spec:2040)/(+index)  
222 xri:(mailto:john.doe@example.com)/(+phone)  
223 xri:=JohnDoe.home/(+email)  
224 xri:=JohnDoe.home/(+email).($v/3)  
225     --cross-references  
226  
227
```

227
228
229
230
231
232
233
234

```
xri:(+flowers.rose)
xri:(//www.example.com/dictionary/flowers/rose)
xri:(http://www.example.com/dictionary/flowers/rose)
--self-references
```

Table 1 also illustrates several examples of internationalized XRIs.

French	xri:@ALaFrançaise/areté
Hebrew	xri://אג.אג.אג.אג/gh/ט' /כ' .html
Kanji	xri:=崎村夏彦/ (+本籍地)

Table 1: Internationalized XRIs.

235
236

1.1.3 URI, URL, URN, and XRI

The evolution and interrelationships of the terms “URI”, “URL”, and “URN” are explained in a report from the Joint W3C/IETF URI Planning Interest Group, “Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations” [RFC3305]. According to section 2.1:

“During the early years of discussion of web identifiers (early to mid 90s), people assumed that an identifier type would be cast into one of two (or possibly more) classes. An identifier might specify the location of a resource (a URL) or its name (a URN), independent of location. Thus a URI was either a URL or a URN.”

This view has since changed, as the report goes on to state in section 2.2:

“Over time, the importance of this additional level of hierarchy seemed to lessen; the view became that an individual scheme did not need to be cast into one of a discrete set of URI types, such as ‘URL’, ‘URN’, ‘URC’, etc. Web-identifier schemes are, in general, URI schemes, as a given URI scheme may define subspaces.”

This conclusion is shared by [RFC2396bis], which states in section 1.1.3:

“An individual [URI] scheme does not need to be classified as being just one of ‘name’ or ‘locator’. Instances of URIs from any given scheme may have the characteristics of names or locators or both, often depending on the persistence and care in the assignment of identifiers by the identifier authority, rather than any quality of the scheme.”

The XRI scheme explicitly implements this philosophy. XRIs can be used either as indirect “names” or direct “locators” for resources, including other XRIs. The XRI scheme also includes syntax for distinguishing whether an XRI is intended only for identification or also for resolution. For more information, see section 2.1.1.4, *Self-References*.

1.2 Design Considerations

The full set of requirements for XRI syntax and resolution is documented in “XRI Requirements and Glossary v1.0 [XRIReqs]. A synopsis of the major design considerations is included here.

1.2.1 Abstraction and Independence

The overarching requirement of the XRI design is that XRI syntax be fully abstract (i.e., independent of resource location, network, application, transport protocol, type, or security method). Although XRI syntax may be extended for specific uses, the generic XRI syntax is

267 designed simply to represent logical associations between resources and therefore to be portable
268 across all networks, directories, domains, and applications.

269 **1.2.2 Persistence and Reassignability**

270 XRI syntax and resolution is designed to express and resolve fully persistent identifiers, fully
271 reassignable identifiers, or any combination of persistent and reassignable identifier segments.

272 **1.2.3 Human-Friendliness and Machine-Friendliness**

273 XRI syntax and resolution is designed to support both human-friendly identifiers (HFIs—those
274 optimized for human readability, memorability, and usability) and machine-friendly identifiers
275 (MFIs—those optimized for machine processing and network efficiency). XRI syntax allows any
276 combination of HFI and MFI components within a single XRI.

277 **1.2.4 Internationalization**

278 XRIs are designed to be rendered in the natural language of the intended user. They therefore
279 employ the Unicode character set [**Unicode**] and provide syntactical support for expressing
280 optional language-dependent context metadata. As a result, XRIs extend the virtues of human
281 readability, memorability, and usability to users of all human languages.

282 **1.2.5 Cross-Context Identification**

283 XRI syntax and resolution is designed to allow the use of an identifier in the context of another
284 identifier (i.e., for an XRI or a URI to be contained within another XRI). Such embedded identifiers
285 are called *cross-references*, and they are vital to XRI extensibility.

286 **1.2.6 Authority, Delegation, and Federation**

287 XRI syntax and resolution are designed to allow any resource to serve as an identifier authority,
288 and for any authority to delegate to any other authority at any level of the path. Thus XRI design
289 imposes no specific delegation model, network topology, or federation structure.

290 **1.2.7 Security and Privacy**

291 XRI syntax and resolution is designed to be adapted to any security model, method, or
292 infrastructure, as well as to any privacy policy or framework. XRIs never require sensitive data,
293 such as passwords or account numbers, to be included in an identifier. If a particular application
294 ever needs to include such data in an XRI, the syntax permits encryption and obfuscation of
295 identifier segments for enhanced security and privacy.

296 **1.2.8 Extensibility**

297 The XRI scheme is designed to provide the same interoperable extensibility for identifiers that
298 XML provides for markup languages. In other words, by design, the XRI scheme should be able
299 to be extended and specialized by various identifier authorities, and these extensions and
300 specializations should be interoperable.

301 **1.3 Terminology and Notation**

302 **1.3.1 Keywords**

303 The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD
304 NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as

305 described in [RFC2119]. When these words are not capitalized in this document, they are meant
306 in their natural language sense.

307 1.3.2 Syntax Notation

308 This specification uses the syntax notation employed in [RFC2396]: Augmented Backus-Naur
309 Form (ABNF), defined in [RFC2234]. Although the ABNF defines syntax in terms of the US-ASCII
310 character encoding, XRI syntax should be interpreted in terms of the character that the ASCII-
311 encoded octet represents, rather than the octet encoding itself, as explained in [RFC2396]. As
312 with URIs, the precise bit-and-byte representation of an XRI on the wire or in a document is
313 dependent upon the character encoding of the protocol used to transport it, or the character set of
314 the document that contains it.

315 The following core ABNF productions are used by this specification as defined by section 6.1 of
316 [RFC2234]: ALPHA, CR, CTL, DIGIT, DQUOTE, HEXDIG, LF, OCTET, and SP. The complete
317 XRI ABNF syntax is collected in Appendix A.

318 To simplify comparison between generic XRI syntax and generic URI syntax, the ABNF
319 productions that are unique to XRIs are shown with light green shading, while those inherited
320 from [RFC2396] or [RFC2396bis] are shown with light yellow shading.

321

322 This is an example of ABNF specific to XRI.

323

324 This is an example of generic URI ABNF from RFC 2396 or 2396bis.

325

326 In addition, productions inherited from the IRI proposal [IRI] are prefixed with the letter “i” just as
327 they are in that document.

328 1.3.3 Glossary

329 The following definitions are central to this specification. Note that this glossary supercedes the
330 glossary in [XRIReqs].

331

332 **Absolute Identifier**

333 An identifier that refers to a resource independent of the current context, i.e., one that
334 establishes a global context. Mutually exclusive with “Relative Identifier.”

335 **Abstract Identifier**

336 An identifier that is not directly resolvable to a resource, but is either:

337 a) a self-reference because it completely represents a non-network resource and is not
338 further resolvable (see “Self-Reference”), or

339 b) an indirect reference to a resource because it must first be resolved to another
340 identifier (either another abstract identifier or a concrete identifier.)

341 A URN as described in [RFC2141] is an example of an abstract identifier. Abstract
342 identifiers provide additional levels of indirection in referencing resources, which can be
343 useful for a variety of purposes, including persistence, equivalence, human-friendliness,
344 and data protection.

345 **Authority (or Identifier Authority)**

346 A resource that assigns identifiers to other resources. Note that in URI syntax as defined
347 in [RFC2396] and [RFC2396bis], the “authority” production refers explicitly to the top-
348 level authority identified by a DNS name or an IP address. Since XRI syntax supports
349 unlimited delegation, the term “authority” can technically refer to an identifier authority at

350 any level. However, in the XRI “authority-path” production (section 2.1.1), it explicitly
351 refers to the top-level identifier authority.

352 **Base Identifier**

353 An absolute identifier that identifies the current context for a relative identifier. See
354 “Relative Identifier.”

355 **Canonical Form**

356 The state of an identifier after applying transformation rules for the purpose of
357 determining equivalence. See also “Normal Form”.

358 **Community (or Identifier Community)**

359 The set of resources that share a common identifier authority, often (but not always) a
360 common root authority. Technically, the set of resources whose identifiers form a directed
361 graph or tree.

362 **Concrete Identifier**

363 An identifier that can be directly resolved to a resource or resource representation, rather
364 than to another identifier. Examples include the MAC address of a networked computer, a
365 phone number that rings directly to a specific device, and a postal address that is not a
366 forwarding address. All concrete identifiers are intended to be resolvable identifiers.
367 Contrast with “Abstract Identifier.”

368 **Context (or Identifier Context)**

369 The resource of which an identifier is an attribute. For example, in the string of identifiers
370 “a/b/c”, the context of the identifier “b” is the resource identified by “a/”, and the context of
371 the identifier “c” is the resource identified by “a/b/”. Since multiple resources may assign
372 an identifier for a target resource, the resource can be said to be identified in multiple
373 contexts. For absolute identifiers, the context is global, i.e., there is a known starting
374 point. For relative identifiers, the context is implicit.

375 **Cross-reference**


376 An identifier assigned in one context that is reused in another context. Cross-references
377 are used primarily to identify logically equivalent resources in different domains or
378 physical locations. For example, a cross-reference may be used to identify the same
379 logical invoice stored in two accounting systems (the originating system and the receiving
380 system), the same logical Web page stored on multiple proxy servers, the same datatype
381 used in multiple databases or XML schemas, or the same abstract concept used in
382 multiple taxonomies or ontologies.

383
384 In XRI syntax, cross-references are syntactically delimited by enclosing them in
385 parentheses. This syntax has a direct analogy in the English language when a word or
386 phrase is enclosed in quotes to indicate that the author is referring to this word or phrase
387 independent of the current context. For example, the phrase “love bird” is quoted in this
388 sentence to indicate its meaning independent of this context.

389 **Delegated Identifier**

390 A multi-segment identifier in which some segments are assigned by different identifier
391 authorities. Mutually exclusive with “Local Identifier.”

392 **Federated Identifier**

393  delegated identifier which spans multiple independent identifier authorities. See also
394 “Delegated Identifier.”

395 **Human-Friendly Identifier (HFI)**

396 An identifier containing words or phrases intended to convey meaning in a specific
397 human language and thus be easy for people to remember and use. Contrast with
398 "Machine-Friendly Identifier."

399 **Identifier**

400 Per **[RFC2396bis]**, anything that "embodies the information required to distinguish what
401 is being identified from all other things within its scope of identification." In UML terms, an
402 identifier is an attribute of a resource (the identifier context) that forms an association with
403 another resource (the identifier target). The general term "identifier" does not specify
404 whether the identifier is abstract or concrete, absolute or relative, persistent or
405 reassignable, human-friendly or machine-friendly, delegated or local, or resolvable or
406 self-referential.

407 **Local Identifier**

408 Any identifier, or any set of segments in a multi-segment identifier, that is assigned by the
409 same identifier authority. Each of these segments is "local" to that authority. Mutually
410 exclusive with "Delegated Identifier."

411 **Machine-Friendly Identifier (MFI)**

412 An identifier containing digits, hex values, or other character sequences optimized for
413 efficient machine indexing, searching, routing, caching, and resolvability. MFIs generally
414 do not contain human semantics. Compare with "Human-Friendly Identifier."

415 **Normal Form**

416 The character-by-character format of an identifier after encoding, escaping, or other
417 character transformation rules have been applied in order to satisfy syntactic
418 requirements. Four normal forms are defined for XRIs—escaped normal form, IRI normal
419 form, anyURI normal form, and URI normal form. See section 2.2.4 for details. See also
420 "Canonical Form".

421 **Persistent Identifier**

422 An identifier that is permanently assigned to a resource and is intended never to be
423 reassigned to another resource, even if the original resource goes off the network, is
424 terminated, or no longer exists. A URN as described in **[RFC2141]** is an example of a
425 persistent identifier. Persistent identifiers tend to be machine-friendly identifiers, since
426 human-friendly identifiers typically reflect human semantic relationships that may change
427 over time. Mutually exclusive with "Reassignable Identifier."

428 **Reassignable Identifier**

429 An identifier that may be reassigned from one resource to another. Example: the domain
430 name "example.com" may be reassigned from ABC Company to XYZ Company, or the
431 email address "john@example.com" may be reassigned from John Smith to John Jones.
432 Reassignable identifiers tend to be human-friendly identifiers because they often
433 represent the potentially transitory mapping of human semantic relationships onto
434 network resources or resource representations. Mutually exclusive with "Persistent
435 Identifier."

436 **Relative Identifier**

437 An identifier that refers to a resource only in relationship to the current context (for
438 example, the current community, the current document, or the current position in a
439 delegated identifier). A relative identifier can be converted into an absolute identifier by
440 combining it with a base identifier (an absolute identifier that identifies the current context
441 of the relative identifier.) See "Base Identifier". Mutually exclusive with "Absolute
442 Identifier."

443 **Resolvable Identifier**

444 An identifier that references a network resource or resource representation and that can
445 be resolved into a network endpoint for communicating with the target resource. Mutually
446 exclusive with "Self-Reference."

447 **Resource**

448 Per [RFC2396bis], "anything that can be named or described." Resources are of two
449 types: network resources (those that are network addressable) and non-network
450 resources (those that exist entirely independent of a network). Network resources are
451 themselves of two types: direct resources (resources which are their entire embodiment)
452 or resource representations (see "Resource Representation").

453 **Resource Representation**

454 A network resource that represents the attributes of another resource. A resource
455 representation may represent either another network resource (such as a machine or an
456 application) or a non-network resource (such as a person, organization, or concept).

457 **Segment**

458 Any syntactically delimited portion of an identifier. In generic URI syntax, all segments
459 after the authority portion are delimited by forward slashes ("/segment1/segment2/..."). In
460 XRI syntax, slash segments can be further subdivided into sub-segments called *dot*
461 *segments* (for reassignable identifiers) and *colon segments* (for persistent identifiers).
462 See section 2.1.2. XRI also supports another type of segment called cross-references,
463 which are enclosed in parentheses. See "Cross-Reference".

464 **Self-Reference (or Self-Referential Identifier)**

465 An identifier which is itself the representation of the resource it references. Self-
466 references are typically used to represent abstract non-network resources (e.g., "love",
467 "Paris", "the planet Jupiter") in contexts where this identifier is not intended to be resolved
468 to a separate network representation of that resource. The primary purpose of self-
469 references is to establish equivalence across contexts (see "Cross-References").
470 Mutually exclusive with "Resolvable Identifier."

471 **Target (or Identifier Target)**

472 The resource referenced by an identifier. A target may be either a network resource
473 (including a resource representation) or a non-network resource.

474 **XRI Reference**

475 A term that includes both absolute and relative XRIs. Used the same way as "URI
476 reference" and "IRI reference". Note that to transform an XRI reference into an XRI, it
477 must be converted into its absolute form.

478

2 Syntax

479

2.1 Syntax Components

480

Generic XRI syntax builds on generic URI syntax. However because it includes syntactic elements and characters outside the range allowed by [RFC2396], this specification does not technically define a new URI scheme. Instead, it follows the example of [IRI] and defines a new identifier scheme, along with a specification for transforming XRIs into generic URIs or IRIs for applications that expect them (see section 2.2.4.3).

485

As with URIs, an XRI may be either absolute or relative.

486

487

```
XRI = absolute-xri / relative-xri
```

488

489

An absolute XRI consists of the scheme name “xri:” followed by the same set of hierarchical components as an absolute URI – authority, path, query, and fragment.

490

491

492

```
absolute-xri = "xri:" global-path
global-path = authority-path [ local-path ]
local-path = "/" relative-path
relative-path = [ xri-segments ] [ "?" xri-query ]
               [ "#" xri-fragment ]
```

493

494

495

496

497

A relative XRI consists of either a *local path* or a *relative path*.

498

499

500

```
relative-xri = local-path / relative-path
```

501

502

Finally, in certain contexts such as cross-references (section 2.1.1.4), the “xri:” scheme name is redundant. These contexts can use the *xri-value* production, which includes all levels of XRI paths.

503

504

505

506

```
xri-value = global-path / local-path / relative-path
```

507

508

2.1.1 Authority

509

XRI syntax supports the same types of authorities as generic URI syntax, called *URI authorities*.

510

In addition, it supports *XRI authorities* that provide two other mechanisms for specifying the global context of an identifier, as defined in section 2.1.1.2.

511

512

513

```
authority-path = URI-authority / XRI-authority
```

514

515

2.1.1.1 URI Authority

516

In the context of an XRI, a URI authority is distinguished by an initial double slash (“//”).

517

518

```
URI-authority = "//" [ userinfo "@" ] host [ ":" port ]
```

519

520 The syntax following this starting delimiter is inherited directly from **[RFC2396bis]**, which
521 simplifies the syntax in **[RFC2396]** and includes support for IPv6 addresses defined in
522 **[RFC2732]**. First, the “userinfo” sub-component permits identifying a user in the context of a host.

523

```
524 userinfo = *( unreserved / escaped / ";" /  
525 ":" / "&" / "=" / "+" / "$" / ", " )
```

526

527 Next, the “host” sub-component has three options for identifying the host: a domain name, an
528 IPv4 address, or an IPv6 literal.

529

```
530 host = [ hostname / IPv4address / IPv6reference ]
```

531

532 Note that the host identifier may be omitted. This is because in generic URI syntax, a default may
533 be defined by the semantics of a particular URI scheme. No default is specified for the XRI
534 scheme; this allows a default to be inherited from the particular protocol used to resolve the XRI.

535 A hostname, after the transformation described in step 4 of section 2.2.4.3, **MUST** meet the rules
536 defined in section 3.2.2 of **[RFC2396]**. The productions for *idomainlabel*, *qualified*, and *hostname*,
537 therefore, have additional restrictions not reflected in the ABNF. See section 2.2.4.3.


538

```
539 hostname = idomainlabel qualified  
540 qualified = *( "." idomainlabel ) [ "." ]  
541 idomainlabel = ( ALPHA / ucschar ) *( alphanum / ucschar / "-" )  
542  
543 alphanum = ALPHA / DIGIT
```

544

```
545 IPv4address = dec-octet "." dec-octet "." dec-octet "." dec-octet  
546 dec-octet = DIGIT ; 0-9  
547 / %x31-39 DIGIT ; 10-99  
548 / "1" 2DIGIT ; 100-199  
549 / "2" %x30-34 DIGIT ; 200-249  
550 / "25" %x30-35 ; 250-255
```

551

552 Support  an IPv6 address literal was added by **[RFC2396bis]** following the syntax originally
553 specified in **[RFC2732]**. Because IPv6 literals use colons as delimiters, they must be
554 encapsulated within square brackets.

555

```
556 IPv6reference = "[" IPv6address "]"  
557 IPv6address = 6( h4 ":" ) ls32  
558 / "::" 5( h4 ":" ) ls32  
559 / [ h4 ] ":" 4( h4 ":" ) ls32  
560 / [ *1( h4 ":" ) h4 ] ":" 3( h4 ":" ) ls32  
561 / [ *2( h4 ":" ) h4 ] ":" 2( h4 ":" ) ls32  
562 / [ *3( h4 ":" ) h4 ] ":" h4 ":" ls32  
563 / [ *4( h4 ":" ) h4 ] ":" ls32  
564 / [ *5( h4 ":" ) h4 ] ":" h4  
565 / [ *6( h4 ":" ) h4 ] ":"  
566 ls32 = ( h4 ":" h4 ) / IPv4address  
567 ; least-significant 32 bits of address  
568 h4 = 1*4HEXDIG
```

569

570 Finally, a host identifier can be followed by an optional port number. Because XRIs are abstract
 571 identifiers, the XRI syntax specification does not define a default port. It is expected that the
 572 default port will be inherited from the resolution protocol, such as the HTTP/HTTPS protocol
 573 specified in section 3. Therefore, if the port is omitted in an XRI, it is undefined.

574

575 `port = *DIGIT`

576

577 2.1.1.2 XRI Authority

578 In addition to the authorities supported in generic URI syntax, XRIs support two other
 579 mechanisms for specifying the global context of an identifier. The first is the global context symbol
 580 (GCS), and the second is the cross-reference (abbreviated in the ABNF as *xref*).

581

582 `XRI-authority = (gcs-char [xri-segment]) / xref-authority`

583

584 2.1.1.3 Global Context Symbols (GCS)

585 To support the abstraction and human-friendly identifier (HFI) requirements, XRIs offer a simple,
 586 compact syntax for indicating the logical global context of an identifier: a single prefix character.

587

588 `gcs-char = "+" / "=" / "@" / "$" / "*" / "!"`

589

590 The global context symbol characters were selected from the set of symbol characters that are
 591 valid in a URI under [\[RFC2396\]](#) to represent the global contexts shown in [Table 2](#)
 592

Symbol Character	Authority Type	Establishes global context for
+	General public	Identifiers for generic concepts for which there is no specific authority, i.e., that are established by public convention. (In the English language, for example, these would be the generic nouns.)
=	Person	Identifiers that represent an individual person.
@	Organization	Identifiers that represent an organization of any kind.
\$	OASIS XRI Metadata Specification	Special identifiers established by the “ <i>XRI Metadata Specification</i> ” [XRIMetadata] for interoperable identifier metadata (e.g., language, version, type, query syntax, etc.).
*	User-relative	Identifiers for which the authority is relative to the current user (“shortcut XRIs”). See section 3.2.6.
!	XRI author	Identifiers used only for human-readable annotations of XRIs (ignored by machine processing.)

593

Table 2: XRI global context symbols.

594 Note that because a global context symbol may precede an xri-segment, and an xri-segment may
 595 start with a cross-reference (below), a global context symbol can be used to express the abstract

596 logical context of a conventional URI authority. For example:
597

```
598 xri:=(http://www.my-website.com)/favorites.html  
599 --expresses that this resource represents an individual
```

600

601 2.1.1.4 Cross-References

602 Cross-references are the primary extensibility mechanism in XRIs. They allow an identifier
603 assigned in one context to be reused in another context, permitting identifiers to be shared across
604 contexts to simplify identifying logically equivalent resources. To syntactically delimit a cross-
605 reference, it is enclosed in parentheses the same way an IPv6 literal is encapsulated in square
606 brackets as specified in [RFC2732] (see section 2.1.1.1). A cross-reference may contain either
607 an XRI value or an absolute URI.

608

```
609 xref-authority = xref ( "." sub-segment / ":" sub-segment )  
610                 * ( "." sub-segment / ":" sub-segment )  
611 xref           = "(" ( xri-value / URI ) ")"
```

612

613 It is important that the value of a cross-reference be syntactically unambiguous, whether it is an
614 absolute URI or one of the various forms of an XRI value. Therefore special attention must be
615 paid to relative XRIs to avoid ambiguity, as discussed in 2.3.3.

616 A cross-reference may appear at any node of any XRI except within a URI authority segment.
617 The use of cross-references as the very first segment in an XRI enables any globally-unique
618 identifier in any URI scheme (e.g., an HTTP URI, mailto URI, URN, etc.) to specify a global
619 authority.

```
621 xri:(mailto:john.doe@example.com)/favorites/home  
622 --example of using a URI as an XRI global authority
```

623

624 2.1.1.5 Self-References

625 Cross-reference syntax is also the means by which an XRI can express that it is not intended for
626 resolution, but only for the purpose of establishing equivalence across contexts. Such an XRI is
627 called a *self-reference*. To express a self-reference, the entire XRI value is enclosed in
628 parentheses—in essence, it becomes a global cross-reference. This is the XRI equivalent of the
629 English language convention of putting a word or phrase in quotes to express that the author is
630 referring to the word or phrase itself and not to its normal meaning. (In linguistics and philosophy,
631 this is called the “use-mention distinction.”) For example:

632

```
633 The term "user-friendly" is used frequently in computing.  
634 --English-language usage of a quoted term  
635  
636 xri:(+user-friendly)  
637 --XRI syntax for expressing a self-reference
```

638

639 2.1.2 Path

640 As with URIs, the XRI path component is a hierarchal sequence of path segments separated by
641 slash (“/”) characters and terminated by the first question-mark (“?”) or number sign (“#”)
642 character, or by the end of the XRI. The key difference is that while a URI path segment is

643 considered opaque by a generic URI processor, an XRI path segment can be parsed by an XRI
644 processor into two types of sub-segments: *dot segments* and *colon segments* after their leading
645 characters (“.” and “:”).

646

```
647 xri-segments = xri-segment-val *( "/" [ xri-segment-val ] )  
648 xri-segment-val = xri-segment / "." / ".."  
649 xri-segment = ( [ "." ] sub-segment / ":" sub-segment )  
650 *( "." sub-segment / ":" sub-segment )  
651 sub-segment = 1*xri-pchar / xref
```

652

653 Dot segments are used to specify *reassignable identifiers*—identifiers that may be reassigned by
654 an identifier authority to represent a different resource at some future date. Colon segments
655 (following the lead of URN syntax in **[RFC2141]**) are used to specify *persistent identifiers*—
656 identifiers that are permanently assigned to a resource and will not be reassigned at a future
657 date. The default is a dot segment, so no leading dot is required if this is the first (or only) sub-
658 segment.

659 Note that for compatability with URI syntax, the ABNF allows two special values of an XRI
660 segment—a single dot and a double dot. These can be used as the leading characters in a
661 relative XRI to indicate its relationship to a base XRI as specified in **[RFC2396]**.

662 Other than these special uses of the dot (“.”) and the colon (“:”) characters, an XRI path segment
663 can contain the same characters as a URI path segment plus the expanded UCS character set
664 (section 2.2.3). If a dot or colon is used, it will be interpreted as a delimiter. If this interpretation is
665 not desired for these characters, or for any other special XRI delimiters, these characters **MUST**
666 be escaped when they appear in the path segment. See section 2.2.4, “Escaped Characters”.

667

```
668 xri-pchar = xri-unreserved / escaped / ";" / "!" / "*" /  
669 "@" / "&" / "=" / "+" / "$" / ", "
```

670

671 With the exception of dot and colon sub-segments, an XRI path segment is considered opaque
672 by generic XRI syntax. As with URIs in general, XRI extensions or generating applications may
673 define special meanings for other URI reserved characters for the purpose of delimiting
674 extension-specific or generator-specific sub-components. For example, section 3.4 of **[RFC2396]**
675 specifies the set of URI reserved characters that can be used within a query segment.

676 2.1.3 Query

677 The XRI query component is identical to the URI query component as described in section 3.4 of
678 **[RFC2396]**, except that it allows the full XRI character range and it may begin with a cross-
679 reference. The latter feature permits the incorporation of XRI metadata describing the query string
680 syntax. See the “*XRI Metadata Specification*” **[XRIMetadata]** for more about query syntax
681 metadata.

682

```
683 xri-query = [ xref ] * ( xri-pchar / "." / ":" / "/" / "?" )
```

684

685 2.1.4 Fragment

686 XRI syntax also supports fragments as described in section 4.1 of **[RFC2396]**, except that it
687 allows the full XRI character range and may begin with a cross-reference.

688

```
689 xri-fragment = [ xref ] * ( xri-pchar / "." / ":" / "/" / "?" )
```

690

691 Since XRI syntax can directly address attributes or secondary representations of a primary
692 resource to any depth, fragments are supported primarily for compatibility with generic URI
693 syntax. XRIs can also employ cross-references to identify media types or other alternative
694 representations of a resource. See section 3.4.4 for an example.

695 2.2 Characters

696 The character set and encoding of an XRI is primarily inherited from generic URI syntax as
697 defined in [RFC2396] and clarified in [RFC2396bis]. However, it also includes the expanded
698 character set defined in [IRI].

699 All XRI characters fall into the same three subsets as URI characters.

700

```
701 xri-characters = xri-reserved / xri-unreserved / escaped
```

702 2.2.1 Character Encoding

703 The basic character encoding of XRI is UTF-8, as recommended by [RFC2718]. When an XRI is
704 presented as a human readable identifier, the representation of the XRI in the underlying
705 document should use the character encoding of the underlying document. However, this string
706 must be converted to UTF-8 before any processing external to the underlying document.

707 Note that not all ASCII sequences can be derived from UTF-8 sequences. A valid XRI character
708 sequence MUST be derivable by unescaping an equivalent UTF-8 sequence. For example, the
709 ASCII sequence '%FC', which would represent U+00FC LATIN SMALL LETTER U WITH
710 DIAERESIS in an iso-8859-1 encoding, when unescaped will not result in a valid UTF-8
711 sequence.

712 2.2.2 Reserved Characters

713 Because additional characters are used to delimit XRI syntax components not present in URIs,
714 the XRI reserved character set is a superset of the URI reserved character set. Specifically, five
715 characters have been added: opening parenthesis ("("), closing parenthesis (")"), dot ("."), asterisk
716 ("*"), and exclamation point ("!").

717

```
718 xri-reserved = "/" / "?" / "#" / "[" / "]" / "(" / ")" / ";" / ":" /  
719 ", " / "." / "&" / "@" / "=" / "+" / "*" / "$" / "!"
```

720

721 If the use of an unescaped XRI reserved character as a data character would cause the
722 interpretation of the XRI to be ambiguous, the character MUST be escaped as per the rules in
723 section 2.2.4, "Escaped Characters", and particularly section 2.2.4.4.

724 2.2.3 Unreserved Characters

725 Aside from the expanded UCS character set for internationalization, the unreserved character set
726 for XRIs is the same as that of URIs after the subtraction of the five characters noted above (all of
727 which are in of the "mark" production of [RFC2396] and [RFC2396bis]).

728

```
729 xri-unreserved = ALPHA / DIGIT / ucschar / xri-mark  
730 xri-mark = "-" / "_" / "~" / "'"
```

731

732 The principal difference between XRI and URI unreserved character sets is the inclusion of the
733 UCS character set.

734

735

736

737

738

739

740

```
ucchar      = %xA0-D7FF / %xF900-FDCF / %xFDF0-FFEF /  
             %x10000-1FFFFD / %x20000-2FFFFD / %x30000-3FFFFD /  
             %x40000-4FFFFD / %x50000-5FFFFD / %x60000-6FFFFD /  
             %x70000-7FFFFD / %x80000-8FFFFD / %x90000-9FFFFD /  
             %xA0000-AFFFFD / %xB0000-BFFFFD / %xC0000-CFFFFD /  
             %xD0000-DFFFFD / %xE1000-EFFFFD
```

741

742

743

744

Escaping unreserved characters in an XRI does not impact what resource is identified by that XRI. However, it may change the result of an XRI comparison (see section 2.4, “Normalization and Comparison”), so unreserved characters should not be escaped unless necessary.

745

2.2.4 Escaped Characters

746

747

748

XRIs follow the same rules for escaping characters as URIs. That is, any data in an XRI MUST be escaped if: a) it does not have a representation using an unreserved character, and b) using a reserved character could cause the XRI to be misinterpreted.

749

750

751

752

753

An XRI thus escaped is said to be in *escaped normal form*. This does not imply that it is necessarily a valid IRI or URI. Rules for converting an XRI into a valid IRI or URI are discussed in section 2.2.4.3. An XRI is in escaped normal form if it is unambiguous per the ABNF provided in this document, but it is a valid IRI or URI only after it is escaped according to the transformation described in section 2.2.4.3.

754

2.2.4.1 Escaped Encoding

755

756

757

XRIs use the same percent-encoding as URIs, described in section 2.4.1 of [RFC2396]. An escaped octet is encoded as a character triplet consisting of the percent character “%” followed by the two hexadecimal digits representing that octet's numeric value.

758

759

```
escaped     = "%" HEXDIG HEXDIG
```

760

761

762

763

764

The uppercase hexadecimal digits “A” through “F” are equivalent to the lowercase digits “a” through “f”, respectively. XRIs that differ only in the case of hexadecimal digits used in escaped octets are equivalent. For consistency, uppercase digits SHOULD be used by XRI generators and normalizers.

765

Note that the % symbol used by itself in an XRI must be escaped as described in section 2.2.5.

766

2.2.4.2 Encoding XRI Metadata

767

768

769

770

771

In some cases, the transformation from an identifier in its native language and display format into an XRI in escaped normal form may lose information that cannot be retained through character escaping. For example, in certain languages displaying the glyph of a UTF-8 encoded character requires additional language and font information not available in UTF-8. The loss of this information during UTF-8 encoding could cause the resulting XRI to be ambiguous.

772

773

774

775

776

777

778

779

Another case is when the normalization or canonicalization rules of a particular identifier authority do not permit the inclusion of whitespace, mixed case letters, or certain punctuation in an XRI segment even when escaped, yet the authority would like to retain this metadata for purposes of presentation. XRI syntax offers an option for encoding this metadata using a cross-reference beginning with the GCS “\$” symbol. As defined in section 2.1.1.3, the top level authority for these identifiers is the “XRI Metadata Specification” [XRIMetadata]. It defines special identifiers for UTF-8 metadata, presentation metadata, and other standard types of identifier metadata together with the rules governing their interpretation.

780 2.2.4.3 Transforming XRIs into IRIs and URIs

781 Although XRIs are intended to be used by applications that understand them natively, it may also
782 be desirable to use them:

- 783 • In contexts that expect a fully-conformant URI reference as defined by **[RFC2396]**.
- 784 • In contexts where there is already a predefined escaping procedure for characters that would
785 otherwise be illegal in a URI under **[RFC2396]**, for example the “anyURI” datatype defined in
786 **[XMLSchema2]**.
- 787 • In contexts where it is desirable to use an Internationalized Resource Identifier as described
788 in **[IRI]**. Note that while **[IRI]** defines the process for converting an IRI to a URI, this
789 conversion differs slightly from the conversion defined for “anyURI” in **[XMLSchema2]** in that
790 it includes an algorithm appropriate for internationalized domain names.

791 This section specifies a progression of steps for transforming an XRI into:

- 792 • A valid IRI (steps 1 – 3 below),
- 793 • A valid anyURI (steps 1 – 4 below), and
- 794 • A valid generic URI (steps 1 – 5 below).

795 Except for transformations specific to XRI syntax, these steps closely follow the algorithm
796 proposed in **[IRI]**.

797 Applications **MUST** transform XRIs to IRIs, anyURIs, or generic URIs using the following steps (or
798 an equivalent process that achieves exactly the same result). These steps assume that the XRI is
799 already in escaped normal form as defined in section 2.2.4.

- 800 1. If the XRI is not encoded in UTF-8, convert the XRI to a sequence of characters encoded
801 in UTF-8, normalized according to Normalization Form C (NFC) as defined in **[UTR15]**.
- 802 2. If necessary, add XRI metadata using cross-references as defined in section 2.2.4.2.
803 Note that the addition of XRI metadata may change the resulting IRI or URI for the
804 purposes of comparison. The significance or insignificance of specific types of XRI
805 metadata is defined in the “XRI Metadata Specification” **[XRIMetadata]**.
- 806 3. Apply the special XRI escaping rules defined in section 2.2.4.4. Note that this step is not
807 idempotent (i.e., each time this step is applied, it may yield different results), so it is very
808 important that implementers not apply this step more than once to avoid changing the
809 semantics of the identifier. At the completion of this step, the escaped XRI may be used
810 as an IRI. This is referred to as *IRI normal form*.
- 811 4. If the XRI has a “hostname” component, replace it with the “hostname” component
812 converted using the “ToASCII” operation defined in section 4.1 of **[RFC3490]**, with the
813 “UseSTD3ASCIIRules” flag set to true and the “AllowUnassigned” flag set to false. At this
814 point the XRI may be used as an anyURI as defined in **[XMLSchema2]** or in a
815 comparable context. This is referred to as *anyURI normal form*.
- 816 5. Replace each character that is disallowed in URI references with escaped triplet(s) as
817 described in section 2.2.4.1, one escaped triplet for each octet in the UTF-8 encoding of
818 the disallowed character. At this point the XRI may be used as a generic URI. This is
819 referred to as *URI normal form*.

820 The form of the XRI that results from each step in this transformation is equivalent to the result of
821 any other step. Applying this conversion does not change the equivalence of the identifier, with
822 the exception of the addition of XRI metadata as discussed in Step 2.

823 In general, an application **SHOULD** use the least escaped version appropriate for the context in
824 which the identifier appears. For example, if the context allows an XRI directly, the identifier
825 **SHOULD** be an XRI in escaped normal form as described in section 2.2.4. If the context allows
826 an IRI but not an XRI, the identifier **SHOULD** be in IRI normal form, and so on.

827 2.2.4.4 Special Escaping Rules for XRI Syntax

828 This section defines special rules for preventing misinterpretation of XRI syntax when an XRI is
829 evaluated by a non-XRI aware processor.

830 The first rule deals with cross-references as explained in section 2.1.1.4. Since a cross-reference
831 contains either a URI or an XRI value (which itself may contain further nested URIs or XRIs), it
832 may include characters that, if not escaped, would cause misinterpretation when the entire XRI is
833 transformed according to the steps in section 2.2.4.3. Consider the following XRI:

834

```
835 xri:@example/(xri:@example2/abc?id=1)
```

836

837 The generic parsing algorithm described in **[RFC2396]** would separate the above XRI into the
838 following components:

839

```
840 scheme = xri  
841 authority = <undefined>  
842 path = @example/(xri:@example2/abc  
843 query = id=1)
```

844

845 The desired separation is:

846

```
847 scheme = xri  
848 authority = <undefined>  
849 path = @example/(xri:@example2?id=1)  
850 query = <undefined>
```

851

852 To avoid this type of misinterpretation, certain characters in a cross-reference must be escaped
853 when transforming an XRI into IRI, anyURI, or URI normal form. In particular, the question mark
854 “?” character must be escaped as “%3F” and the number sign “#” character must be escaped as
855 “%28”.

856 Following this rule, the above example would be expressed as:

857

```
858 xri:@example/(xri:@example2%3Fid=1)
```

859

860 In addition, the slash “/” character in a cross-reference may also be misinterpreted by a non-XRI
861 aware processor. Consider:

862

```
863 xri://example.com/(@example/abc)
```

864

865 If this were used as a base URI as defined in section 5 of **[RFC2396]**, the algorithm described in
866 section 5.2 of **[RFC2396]** would append a relative-path reference to:

867

```
868 xri://example.com/(@example/
```

869

870 instead of the intended:

871

```
872 xri://example.com/
```

873

874 This is because the algorithm is defined in terms of the last (right-most) slash character. This
875 problem is avoided by escaping slashes within cross-references as "%2F". Following this rule, the
876 above example would now be expressed as:

877

```
878 xri://example.com/(@example%2Fabc)
```

879

880 Ambiguity is also possible if an XRI in escaped normal form contains characters that have been
881 escaped to indicate that they should not be interpreted in their normal syntactical sense. For
882 example, consider the following XRI in escaped normal form:

883

```
884 xri://example.com/(@example/abc%2Fd/ef)
```

885

886 This slash character between "c" and "d" is escaped to show that it's not a syntactical element of
887 the XRI, i.e., that it should be interpreted literally and not as a path separator. To preserve this
888 type of distinction when converting an XRI to an IRI or URI, the percent "%" character must be
889 escaped as "%25". Following this rule, the above example fully converted would be:

890

```
891 xri://example.com/(@example%2Fabc%252Fd%2Fef)
```

892

893 To summarize, the following four special escaping rules **MUST** be applied during Step 3 of
894 section 2.2.4.3. Before applying these rules, the XRI **MUST** be in escaped normal form and all
895 URIs in cross-references **MUST** be in an escaped form appropriate to their schemes.

- 896 1. Escape all percent "%" characters as "%25" across the entire XRI.
- 897 2. Escape all number sign "#" characters that appear within a cross-reference as "%23".
- 898 3. Escape all question mark "?" characters that appear within a cross-reference as "%3F".
- 899 4. Escape all slash "/" characters that appear within a cross-reference as "%2F".

900 2.2.4.5 Transforming URIs and IRIs Back into XRIs

901 Transformation of an XRI in IRI, anyURI, or URI normal form into an XRI in escaped normal form
902 **MUST** use the following steps (or an equivalent process that achieves the same result). Except
903 for the steps specific to XRI syntax, this procedure very closely follows the algorithm defined in
904 **[IRI]**.

905 If the XRI is in URI normal form, perform this sequence of steps:

- 906 1. If the identifier is not encoded in US-ASCII, convert it to a sequence of octets in US-
907 ASCII.
- 908 2. If the identifier has a "hostname" component, replace it with the UTF-8 encoded
909 "hostname" component converted using the "ToUnicode" operation defined in section 4.2
910 of **[RFC3490]**, with the "UseSTD3ASCIIRules" flag set to true and the "AllowUnassigned"
911 flag set to false.
- 912 3. Convert all escaped characters (as defined in section 2.2.4) with their corresponding
913 octets, except for the percent "%" character, those characters in the "reserved"
914 production of **[RFC2396]** and US-ASCII characters disallowed in URIs by section 2.4.3 of
915 **[RFC2396]**.
- 916 4. Re-escape any octet produced in step 3 that is not part of a strictly legal UTF-8 octet
917 sequence.
- 918 5. Perform the following special conversions for XRI syntax:
919 a. Convert all escaped slash "/" characters to their corresponding octets.

- 920 b. Convert all escaped question mark “?” characters to their corresponding octets.
 921 c. Convert all escaped number sign “#” characters to their corresponding octets.
 922 d. Convert all escaped percent “%” characters to their corresponding octets.
- 923 6. Encode the resulting sequence in UTF-8 (except for that portion already converted by
 924 step 3).
- 925 If the XRI is in anyURI normal form, perform this sequence of steps:
- 926 1. If the XRI is not encoded in UTF-8, convert the XRI to a sequence of characters encoded
 927 in UTF-8, normalized according to Normalization Form C (NFC) as defined in [UTR15].
 928 2. Perform Step 2 above.
 929 3. Perform Step 5 above.
- 930 If the XRI is in IRI normal form, perform the same steps as with an XRI in anyURI normal form,
 931 except skip the second step.

932 2.2.5 Excluded Characters

933 XRI syntax excludes the same characters as URI syntax for the same reasons as described in
 934 section 2.5 of [RFC2396] and [RFC2396bis]. Data octets corresponding to these characters
 935 MUST be escaped in order to be represented within an XRI.

936

```

937   excluded       = invisible / delims / unwise
938   invisible     = CTL / SP / %x80-FF
939   delims        = "<" / ">" / "%" / DQUOTE
940   unwise        = "{" / "}" / "|" / "\" / "^" / "`"
  
```

941

942 As with IRIs, infrastructure responsible for accepting or presenting XRIs MAY deal with
 943 characters in the "excluded" set above, escaping them on input and/or unescaping them prior to
 944 rendering as described in section 2.2.4. A string that contains these characters in an unescaped
 945 form, however, is not technically a valid XRI.

946 Note that in certain contexts, presenting "space" or other whitespace characters in unescaped
 947 form may present special risks for several reasons. First, it is often difficult to visually determine
 948 the number of spaces or other characters composing a block of whitespace, leading to
 949 transcription errors. Second, the space character is often used to delimit an XRI, so unescaped
 950 spaces or whitespace characters can make it difficult or impossible to determine where the
 951 identifier ends. Finally, unescaped spaces or whitespace can be used to maliciously construct
 952 subtly different identifiers intended to mislead the reader. For these reasons, unescaped spaces
 953 or whitespace characters SHOULD be avoided in presentation.

954 [IRI] provides the following guidance concerning other characters that should be avoided. This
 955 guidance applies to XRIs as well.

956 The UCS also contains many areas of characters for which there are strong
 957 visual look-alikes. Because of the likelihood of transcription errors, these also
 958 should be avoided in IRIs. These include the full-width equivalents of ASCII
 959 characters, half-width Katakana characters for Japanese, and many others. This
 960 also includes many look-alikes of "space", "delims", and "unwise", characters
 961 excluded in [RFC3491].

962 Additional information is available from [UniXML]. [UniXML] is written in the
 963 context of running text rather than in the context of identifiers. Nevertheless, it
 964 discusses many of the categories of characters not appropriate for IRIs.

965 **2.3 Relative XRI References**

966 The authority component of an XRI may be either a URI-authority (section 2.1.1.1) or an XRI-
967 authority (section 2.1.1.2). In this section, “authority” should be understood as defined by section
968 2.1.1 of this specification and not in the narrower sense of section 3.2 of **[RFC2396]**.

969 For a relative XRI reference whose base XRI contains an authority component matching the URI-
970 authority production, the rules for resolving relative references defined in section 5.2 of
971 **[RFC2396]** apply. However, for a relative XRI reference whose base XRI contains an authority
972 component matching the XRI-authority production, the rules defined in section 5.2 of **[RFC2396]**
973 need modification because an XRI authority is considered opaque by generic URI syntax.

974 The following sections, therefore, define the process for resolving a relative XRI reference into a
975 fully qualified XRI regardless of the type of authority involved.

976 **2.3.1 Establishing a Base XRI**

977 A base XRI is established according to the rules defined in section 5.1 of **[RFC2396]**. Applying
978 these rules, however, may require the conversion of the XRI into URI normal form as described in
979 section 2.2.4.3. Once in URI normal form, there is no difference between establishing a base XRI
980 and establishing the base of any URI.

981 **2.3.2 Obtaining the Referenced XRI**

982 Section 5.2 of **[RFC2396]** describes rules for resolving relative references to absolute forms of
983 URIs. For XRIs, these rules apply with the following modifications:

- 984 • In step 1, the XRI reference is parsed using an XRI aware parser such that the “authority”
985 component is interpreted as the “authority-path” production defined in section 2.1.1 of this
986 specification.
- 987 • Step 4 states, “If the ‘authority’ component is defined, then the reference is a ‘network-path’
988 and we skip to step 7”. For XRIs, the presence of an “authority” component does not imply
989 that the reference is a “network-path” as defined by **[RFC2396]** because it may be an “XRI-
990 authority” component. However, the instruction to skip to step 7 is still valid for XRIs. In other
991 words, the processing instruction is correct, but the inference as to the type of reference is
992 invalid.
- 993 • In step 4, the base XRI is parsed using an XRI-aware parser such that the “authority”
994 component is interpreted as the “authority-path” production defined in section 2.1.1 of this
995 specification.
- 996 • In step 7, the block that reads:

```
997     if authority is defined then  
998         append "/" to result  
999         append authority to result
```

1000 is replaced by

```
1001     if authority is defined then  
1002         if type-of(authority) == URI-authority  
1003             append "/" to result  
1004             append authority to result
```

1005

1006 It is important to note that the algorithm described in section 5.2 of **[RFC2396]** will generally
1007 produce incorrect results when applied to relative XRI references where the authority component
1008 matches the “XRI-authority” production. This type of relative XRI reference, therefore, should only
1009 be used in contexts in which the algorithm specified in this section is known to be employed.

1010 The following are examples of resolving relative XRI references. These examples closely follow
1011 the examples for resolving relative references in URIs in appendix C of **[RFC2396]**. Starting with a
1012 base XRI of:

1013 `xri:@a.b.c/d.e/f;p?q`

1014 the following relative XRIs would be resolved as shown:

1015	<code>.g:h</code>	=	<code>xri:@a.b.c/d.e/.g:h</code>
1016	<code>./g:h</code>	=	<code>xri:@a.b.c/d.e/g:h</code>
1017	<code>g:h</code>	=	<code>g:h</code> (see section 2.3.3 below)
1018	<code>g</code>	=	<code>xri:@a.b.c/d.e/g</code>
1019	<code>./g</code>	=	<code>xri:@a.b.c/d.e/g</code>
1020	<code>g/</code>	=	<code>xri:@a.b.c/d.e/g/</code>
1021	<code>/g</code>	=	<code>xri:@a.b.c/g</code>
1022	<code>?y</code>	=	<code>xri:@a.b.c/d.e/?y</code>
1023	<code>g?y</code>	=	<code>xri:@a.b.c/d.e/g?y</code>
1024	<code>#s</code>	=	(current document)#s
1025	<code>g#s</code>	=	<code>xri:@a.b.c/d.e/g#s</code>
1026	<code>g?y#s</code>	=	<code>xri:@a.b.c/d.e/g?y#s</code>
1027	<code>;x</code>	=	<code>xri:@a.b.c/d.e;/x</code>
1028	<code>g;x</code>	=	<code>xri:@a.b.c/d.e/g;x</code>
1029	<code>g;x?y#s</code>	=	<code>xri:@a.b.c/d.e/g;x?y#s</code>
1030	<code>.</code>	=	<code>xri:@a.b.c/d.e/</code>
1031	<code>./</code>	=	<code>xri:@a.b.c/d.e/</code>
1032	<code>..</code>	=	<code>xri:@a.b.c/</code>
1033	<code>../</code>	=	<code>xri:@a.b.c/</code>
1034	<code>../g</code>	=	<code>xri:@a.b.c/g</code>

1035 As with URIs, the “..” syntax cannot be used to change the authority component of an XRI.

1036	<code>../..</code>	=	<code>xri:@a.b.c/.../</code>
1037	<code>../.../</code>	=	<code>xri:@a.b.c/.../</code>
1038	<code>../.../g</code>	=	<code>xri:@a.b.c/.../g</code>

1039 **2.3.3 Leading Segments Containing a Colon**

1040 **[RFC2396]** points out that relative URI references with an initial segment containing a colon may
1041 be subject to two interpretations:

1042 Authors should be aware that a path segment which contains a colon character
1043 cannot be used as the first segment of a relative URI path (e.g., “this:that”),
1044 because it would be mistaken for a scheme name.

1045 It is therefore necessary to precede such segments with other segments (e.g.,
1046 “./this:that”) in order for them to be referenced as a relative path.

1047 Relative XRI references can be similarly misinterpreted. Therefore if any segment prior to the first
1048 forward slash (“/”) character in a relative XRI reference contains a colon, the relative XRI
1049 reference must be rewritten to begin either with a “.” or a “./”. Thus, “foo:bar” becomes “.foo:bar”
1050 or “./foo:bar” and “foo.bar:baz” becomes “.foo.bar:baz” or “./foo.bar:baz”. Note that by the rules of
1051 sections 2.3.2 and 2.4.3, this transformation does not affect equivalence.

1052 **2.4 Normalization and Comparison**

1053 In general, the normalization and comparison rules for generic URIs specified in **[RFC2396]** apply
1054 to XRIs in URI normal form, namely that the scheme and hostname are case insensitive. This
1055 section describes a number of additional XRI-specific rules for normalization and comparison.
1056 To reduce the requirements imposed upon a minimally conforming processor, the majority of
1057 these rules are RECOMMENDED rather than REQUIRED. An implementation that fails to
1058 observe them, however, may frequently treat two XRIs as non-equal when in fact they are equal.

1059 In addition to these rules, Section 6 of **[RFC2396bis]** offers advice on more aggressive strategies
1060 for normalization. Although entirely non-normative, implementers may find this information useful
1061 in developing a strategy for establishing equivalence, particularly with respect to XRIs containing
1062 cross-references to URIs.

1063 Finally, each application that uses XRIs MAY define additional equivalence rules as appropriate.
1064 Due to the level of abstraction XRIs provide, such higher-order equivalence rules may be based
1065 on indirect comparisons or specified XRI-to-XRI mappings (for example, mappings of
1066 reassignable XRIs to persistent XRIs).

1067 **2.4.1 Case**

1068 The following rules regarding case sensitivity SHOULD be applied in XRI comparisons.

- 1069 • Comparison of the scheme component of XRIs and all URIs used as cross-references is
1070 case-insensitive.
- 1071 • Comparison of URI authority components as defined in section 2.1.1.1 is case-insensitive as
1072 defined in **[RFC2396]**.
- 1073 • Comparison of XRI authority components as defined in section 2.1.1.2 is case-insensitive.
1074 Specifically, because an XRI authority component can contain a wide range of Unicode
1075 characters, two XRI authority components are equivalent if they match according to the
1076 compatibility caseless match operation defined in section 3.13 of **[Unicode]** after applying
1077 steps 1 and 3 of the transformation described in section 2.2.4.3.
- 1078 • As specified in section 2.2.4.1, comparison of percent-encoded characters is case-insensitive
1079 for the hexadecimal digits “A” through “F”.

1080 **2.4.2 Encoding, Escaping, and Transformations**

- 1081 • Two XRIs MUST be considered equivalent if they are character-for-character equivalent.
1082 Therefore, they are also equivalent if they are byte-for-byte equivalent and use the same
1083 character encoding.
- 1084 • Two XRIs that differ only in escaped unreserved characters SHOULD be considered
1085 equivalent. If one XRI escapes one or more unreserved characters, and another XRI is
1086 different only in that the same characters are not escaped, they are equivalent.
- 1087 • All forms of an XRI during the transformation process described in section 2.2.4.3 SHOULD
1088 be considered equivalent, assuming the same XRI metadata is inserted as described in
1089 section 2.2.4.2.

1090 **2.4.3 Optional Syntax**

- 1091 • An xri-segment (section 2.1.2) that omits the optional leading dot (“.”) is equivalent to the
1092 same xri-segment prefixed with the leading dot. For example the segment “/foo.bar” is
1093 equivalent to the segment “./foo.bar”.
- 1094 • A cross-reference (section 2.1.1.4) that begins with the GCS symbol for annotations (“!”) AND
1095 the delimiter that precedes the cross-reference SHOULD be ignored entirely for purposes of
1096 comparison. For example, “xri:@:A6B4.(!www.example.org):5E32” is equivalent to
1097 “xri:@:A6B4:5E32”. Note that because XRI annotations are explicitly designed to be ignored
1098 by XRI processors, failure to observe this rule will cause XRIs that are intended to be
1099 equivalent to be incorrectly evaluated.

1100 **2.4.4 Cross-References**

- 1101 • If an XRI contains a cross-reference, the rules in this section SHOULD be applied recursively
1102 to each cross-reference. For example, the following two XRIs should be considered
1103 equivalent:

1104

1105

1106

```
xri:@example/(+example/(+foo))
xri:@example/(+Example/(+FOO))
```

1107

1108

- From the standpoint of XRI syntax, all cross-references beginning with the GCS “\$” symbol SHOULD be considered significant unless stated otherwise in the “XRI Metadata Specification” [XRIMetadata]. See section 2.2.4.2.

1109

1110

1111

2.4.5 Canonicalization

1112

In general, XRIs do not have a single canonical form. This is particularly true for XRIs that contain URI cross-references, since many URI schemes, including the HTTP scheme, do not define a canonical form. Additionally, the authority for a particular segment of an XRI may define its own rules with respect to case-sensitivity, optional or implicit syntax, etc., making canonicalization of those segments outside the scope of this specification.

1113

1114

1115

1116

1117

Nevertheless it is valuable to define guidelines for making XRIs reasonably canonical. XRIs that follow these guidelines will be more consistent in presentation, simpler to process, less prone to false-negative comparisons, and more easily cached. To that end, unless there is a compelling reason to do otherwise, XRIs should be provided them in a form in which:

1118

1119

1120

1121

- The optional xri scheme is added,
- The scheme is provided in lowercase,
- The authority component is provided in lowercase,
- Percent-escaping uses uppercase A through F,
- If optional, the leading dot in xri-segments is omitted,
- Unnecessary escaping is removed,
- ./ and ../ are absent in absolute XRIs, and
- Cross-references are reasonably canonical with respect to their schemes.

1122

1123

1124

1125

1126

1127

1128

1129

1130

Table 3 illustrates the application of these rules. Although the XRIs in the first and second columns are equivalent, the form in the second column is recommended.

1131

1132

Avoid	Recommended	Comment
@example	xri:@example	Add optional scheme
XRI:@example	xri:@example	Lowercase scheme
xri:@Example	xri:@example	Lowercase authority
xri:@example%2f	xri:@example%2F	Uppercase percent escaping
xri:@example/.abc	xri:@example/abc	Remove optional leading dot
xri:@ex%61mple	xri:@example	Remove unnecessary escaping
xri:@example/./abc	xri:@example/abc	Avoid ./ and ../ in absolute XRIs

1133

Table 3: Examples of XRI canonicalization recommendations.

1134

3 Resolution

1135

3.1 Introduction

1136

XRI resolution is the process of dereferencing an XRI to a network endpoint in order to communicate with the resource identified by the XRI. Because XRIs may be used across a wide variety of communities and applications, including as database keys, filenames, directory keys, object IDs, and XML IDs, no single resolution mechanism may be appropriate for all XRIs.

1139

However, in the interest of promoting interoperability, this specification defines a simple, flexible resolution protocol that relies exclusively on HTTP/HTTPS for network transport.

1140

1141

Identifier management policies are defined on a community-by-community basis. With XRIs, the authoritative community is specified by the authority segment of the XRI (section 2.1.1). When a community chooses to create a new identifier authority, it SHOULD define a policy for assigning and managing identifiers under this authority. Furthermore, it SHOULD define what resolution protocol(s) can be used for resolving identifiers assigned by the authority.

1142

1143

1144

1145

1146

1147

3.1.1 Assumptions

1148

This resolution protocol makes several minimal assumptions about the XRIs being resolved:

1149

- The endpoints representing the top-level authority for any globally unique XRI are identified with the “URI-authority” or “XRI-authority” segment of the XRI as defined in section 2.1.1. If the endpoint identified by an XRI authority begins with a cross-reference to another URI scheme (for example, the URN scheme), this cross-reference must be resolved by that XRI authority. (Although other protocols could be specified by that XRI authority to resolve such URI cross-references, such protocols are outside the scope of this specification.)
- Only absolute XRIs are resolved using this protocol. To resolve a relative XRI, it must be converted into an absolute XRI using the procedure in section 2.3.
- The XRI being resolved has been converted into URI normal form, following the rules in section 2.2.4.3.
- Data or metadata associated with a single XRI may be retrieved or manipulated by multiple protocols at multiple endpoints.
- Each endpoint may present a different subset, type, or representation of data or metadata associated with the identified resource.

1150

1151

1152

1153

1154

1155

1156

1157

1158

1159

1160

1161

1162

1163

1164

3.1.2 Phases of Resolution

1165

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

1166

1167

- Authority resolution
- Local access

1168

1169

Authority resolution is the process of finding the endpoint or endpoints representing the top-level identifier authority for the XRI. The result of authority resolution is a list of local access endpoints identified by one or more URIs and supporting at least one local access protocol. An XRI resolver chooses one of these endpoints and then accesses it using the desired local access protocol.

1170

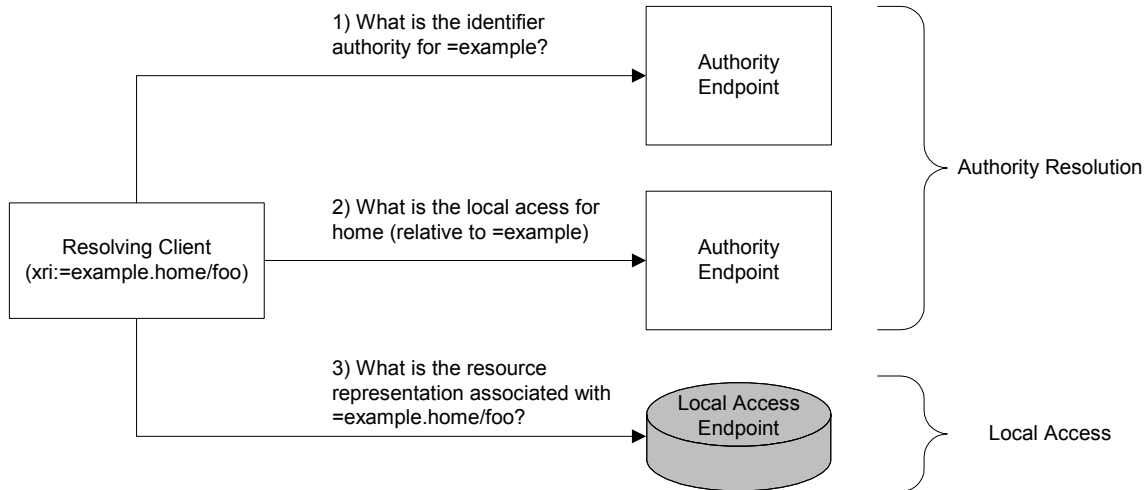
1171

1172

1173

[Figure 1](#) illustrates these two phases of XRI resolution:

1174



1175
1176

Figure 1: Phases of Resolution

1177 **3.1.3 URI vs. XRI Authorities**

1178 As described in sections 2.1.1.1 and 2.1.1.2, URI and XRI authorities have different syntactic
 1179 structures, partially due to the higher layer of abstraction represented by XRI authorities. For this
 1180 reason, XRI authorities are resolved sub-segment by sub-segment as described in section 3.2,
 1181 while URI authorities, since they are based on DNS names or IP addresses, are resolved by
 1182 transforming the XRI to a HTTP URI as described in section 3.3.

1183 **3.1.4 XRI Metadata Reserved for XRI Resolution**

1184 As defined in section 2.1.1.3, the GCS symbol "\$" is reserved for XRI metadata, i.e., special
 1185 identifiers assigned by this specification or the "XRI Metadata Specification" [XRIMetadata] to
 1186 describe or resolve other identifiers.
 1187 Within the "\$" namespace, the identifier "\$r" is reserved for identifiers assigned by this resolution
 1188 specification. [Table 4](#) summarizes these identifiers.
 1189

Identifier	Use	See Section
\$r.s	XML namespace for XRI resolution schemas	3.2.2
\$r.a	Namespace for local access protocol types	3.4.1
\$r.t	Namespace for resource representation types	3.4.4

1190 Table 4: Special identifiers reserved for XRI resolution.

1191 **3.2 XRI Authority Resolution**

1192 **3.2.1 Overview**

1193 XRI authority resolution is an iterative process that resolves the qualified sub-segments within the
 1194 XRI authority segment from left to right. A qualified sub-segment is either: a) a global context
 1195 symbol as defined in section 2.1.1.3, or b) a sub-segment as defined in section 2.1.2 *together*

1196 *with its preceding syntax delimiter* (“.” or “:”). Note that in the latter case a qualified sub-segment
1197 always includes the syntax delimiter even if it was optionally omitted in the original XRI.

1198 Each **qualified** sub-segment is resolved in the context of the **qualified** sub-segment immediately to
1199 the left. The first (or leftmost) **qualified** sub-segment specifies the root of the identifier community.
1200 Each XRI community provides, by definition, one or more network endpoints (HTTP or HTTPS
1201 URIs) that answer resolution requests at the root level. This starting point is further discussed in
1202 section 3.2.3.

1203 After the starting HTTP/HTTPS URI is determined, the resolution process proceeds to the next
1204 **qualified** sub-segment to the right. Each **qualified** sub-segment is resolved to an XRI Descriptor
1205 as defined in section 3.2.2. This XML instance provides the data and metadata necessary to
1206 construct the URI for the next XRI authority as described in section 3.2.4. Once the final XRI
1207 authority is reached, the XRI Descriptor provides the available local access service protocol(s) as
1208 discussed in section 3.4. In addition, the XRI Descriptor can provide a mapping of other XRIs
1209 equivalent to the resolved XRI.

1210 All three options—next authority, local access, or mapping—may be available at every iteration.
1211 For example, the XRI authority identifier “@a.b.c” may be the prefix to another XRI authority with
1212 the XRI “@a.b.c.d”. Or “@a.b.c” may be a local access endpoint itself, in which case its XRI
1213 Descriptor will contain references to local access services. Finally, this XRI Descriptor can also
1214 assert that the identifier “xri:@a.b.c” maps to the identifier “xri:@:1:2:3” in order to provide
1215 resolvers or caches with an equivalent persistent XRI.

1216 3.2.2 XRI Descriptors

1217 To provide a straightforward, flexible resolution mechanism, XRI authority endpoints are
1218 described using a simple XML document with a very flexible content model. Its purpose is only to
1219 provide the data and metadata necessary to support delegated resolution and access of XRI-
1220 identified authorities and resources.


1221 The formal XML Schema definition of an XRI Descriptor is provided in Appendix B. The following
1222 example illustrates the fields defined in this schema:
1223

```
1224 <XRIDescriptor xmlns="xri:$r.s/XRIDescriptor">  
1225   <Resolved>.foo</Resolved>  
1226   <XRIAuthority>  
1227     <URI>http://xri.example.com</URI>  
1228     <URI>https://xri.example.com</URI>  
1229   </XRIAuthority>  
1230   <LocalAccess>  
1231     <Service> xri:$r.a/X2R</Service>  
1232     <Type>application/rddl+xml</Type>  
1233     <URI>http://xri.example.com</URI>  
1234   </LocalAccess>  
1235   <LocalAccess>  
1236     <Service> xri:$r.a/X2R</Service>  
1237     <Type>image/jpeg</Type>  
1238     <URI>http://pictures.xri.example.com</URI>  
1239   </LocalAccess>  
1240   <Mapping>xri:@:1:2:3</Mapping>  
1241 </XRIDescriptor>
```

1243 All schema elements are in the XML namespace “xri:\$r.s/XRIDescriptor”. Following are the
1244 elements and attributes that comprise the XRIDescriptor document type:

1245 /XRIDescriptor

1246 Required. The outer element of the XRIDescriptor document.

- 1247 **/XRIDescriptor/Expires**
- 1248 0 or 1. The UTC time at which this document MUST no longer be relied upon. A resolver
1249 MAY discard this document before the time indicated in this result. If the HTTP transport
1250 caching semantics specify an expiry time which is earlier than the time expressed in this
1251 attribute, then the “XRIDescriptor” document MUST no longer be relied upon after the
1252 expiry time declared in the HTTP headers per section 13.2 of [RFC2616].
- 1253 **/XRIDescriptor/Resolved**
- 1254 0 or 1. Expresses the **qualified** sub-segment whose resolution results in this
1255 XRIDescriptor document. This field can be used in conjunction with Digital Signatures to
1256 provide secure resolution (functionality that is not specified in this document, but which
1257 will be part of a future deliverable of the OASIS XRI TC). This field may also be useful for
1258 debugging or auditing purposes.
- 1259 **/XRIDescriptor/XRIAuthority**
- 1260 0 or 1. Indicates the next XRI authority to query if the resolved **qualified** sub-segment
1261 represents another XRI authority. If the resolved **qualified** sub-segment does not identify
1262 another XRI authority, but rather a resource in the context of the current XRI authority,
1263 then this element may not be present.
- 1264 **/XRIDescriptor/XRIAuthority/URI**
- 1265 1 or more. Required if “XRIAuthority” element is present. Indicates the transport level URI
1266 where the next XRI authority can be contacted. Required by this specification to be an
1267 HTTP or HTTPS URI. Future extensions may use other transport protocols.
- 1268 **/XRIDescriptor/LocalAccess**
- 1269 0 or more. Indicates that the resolved **qualified** sub-segment specifies a identifier
1270 authority where local access service is available.
- 1271 **/XRIDescriptor/LocalAccess/Service**
- 1272 0 or 1. Indicates the type of local access service. The service type is specified by a URI
1273 (including the URI normal form of an XRI). This specification defines one service: “X2R”
1274  which is identified with the URI “xri:\$r.a/X2R” (see section 3.4.1.) An X2R service
1275 converts the XRI into a digital representation of that resource. No more specific
1276 semantics are defined. If this element is absent, then the service associated with this
1277 local access endpoint is X2R.
- 1278 **/XRIDescriptor/LocalAccess/Type**
- 1279 0 or more. The media type of content available at this service. If this element is not
1280 present, then no assumption can be made about the type of data available at this
1281 endpoint. The content of this attribute must be of the form of a media type as defined in
1282 [RFC2046]. This element may appear multiple times to indicate multiple media types
1283 available through this local access service.
- 1284 **/XRIDescriptor/LocalAccess/URI**
- 1285 1 or more. Required if “LocalAccess” element is present. Indicates the transport-level URI
1286 at which the local access service can be requested.
- 1287 **/XRIDescriptor/Mapping**
- 1288 0 or more. Represents an equivalent XRI to the described XRI. Must be an absolute XRI
1289 (“absolute-xri” in the ABNF, section 2.1.)
- 1290 XRI mapping may be used, for example, to assert that a XRI authority known by a
1291 reassignable XRI may also be known by one or more persistent XRIs, or by a different
1292 reassignable XRI than the one which is being resolved. Both cases may be particularly
1293 useful in populating or querying a cache.
1294

1295 XRI Descriptor documents have an “open schema” that allows other elements and attributes from
 1296 other namespaces to be added throughout. These points of extensibility can be used to deploy
 1297 new identifier authority or local access resolution schemes.

1298 Another possible extension is the attachment of XML Digital Signatures and SAML assertions to
 1299 support secure resolution. The current protocol does not specify such security features, although
 1300 a secure resolution protocol is a future deliverable of the OASIS XRI TC.

1301 3.2.3 Initiating Resolution

1302 With an XRI authority, the first qualified sub-segment corresponding to the community root may
 1303 be a global context symbol (GCS) or a cross-reference. In either case, the associated community
 1304 must have published an XRI Descriptor that contains one or more HTTP or HTTPS URIs
 1305 declaring the root resolvers for the community. This XRI Descriptor is known *a priori* and is part of
 1306 the configuration of a resolver, not unlike the configuration of root DNS servers in a DNS resolver.

1307 It is important to note that if the sub-segment following the GCS does not begin with a colon
 1308 (meaning it is not a persistent identifier), then a dot is implied, and a dot must be added when
 1309 constructing the qualified sub-segment. [Table 5](#) and [Table 6](#) demonstrate the
 1310 parsing of such a sub-segment in the case of a GCS and a cross-reference, respectively.

1311

XRI	xri:@example.internal/foo
XRI Authority	@example.internal
Identifier Community	@
First <u>Qualified</u> Sub-segment Resolved	.example

1312 Table 5: Parsing the first sub-segment of an XRI that begins with a global context symbol.

1313

XRI	xri:(http://www.example.com).internal/foo
XRI Authority	(http://www.example.com).internal
Identifier Community	(http://www.example.com)
First <u>Qualified</u> Sub-segment Resolved	.internal

1314 Table 6: Parsing the first sub-segment of an XRI that begins with a cross-reference.

1315

1316 3.2.4 Iterating Resolution

1317 Once the XRI Descriptor representing the community root authority is known, the resolution
 1318 process begins an iteration by constructing the Next Resolution URI. With each iteration, the Next
 1319 Resolution URI is constructed from the same two parts:

- 1320 1. The XRI Authority URI extracted from the XRI Descriptor corresponding to the current
 1321 context,
- 1322 2. The next qualified sub-segment, which always begins with an XRI syntax delimiter (“.” or
 1323 “:”) (see the clarification regarding cross-references in section 3.2.6).
 1324

1325 The URI which forms the base of the Next Resolution URI is the value of a URI element found at
 1326 [XPath element path /XRIDescriptor/XRIAuthority/URI](#) in the XRI Descriptor. If the path portion
 1327 of this URI does not end with a “/” character, one must be appended before proceeding. The URI

1328 normal form (section 2.2.4.3) of the qualified sub-segment being resolved is then appended to the
1329 path portion of the URI. As noted above, if there is no separator character preceding the sub-
1330 segment, a "." MUST be added when creating the qualified sub-segment.

1331 For example, when resolving the "c" sub-segment of "xri:@a.b.c", if the XRI Authority URI
1332 resulting from the resolution of "xri:@a.b" is "http://example.com/xri-authority/", then the Next
1333 Resolution URI is the concatenation of "http://example.com/xri-authority/" with ".c", yielding
1334 "http://example.com/xri-authority/.c". An HTTP request is made to this URI, and the next XRI
1335 Descriptor for the context "xri:@a.b.c" is retrieved.

1336 Construction of the Next Resolution URI is more formally described in this pseudo-code:
1337

```
1338 xa-uri = xri-authority-uri  
1339  
1340 if (path portion of xa-uri doesn't end in "/"):  
1341     append "/" to path portion of xa-uri  
1342  
1343 if (current-sub-segment isn't preceded with "." or ":" separator):  
1344     xa-uri = xa-uri + "."  
1345 else:  
1346     xa-uri = xa-uri + separator  
1347  
1348 xa-uri = append uri-escape(sub-segment) to path portion of xa-uri
```

1349

1350 Once the Next Resolution URI is constructed, an HTTP or HTTPS GET request is made using
1351 this URI. Each GET request results in a 2XX or 304 HTTP response. The HTTP/HTTPS response
1352 should either contain the next XRI Descriptor or, with a 304 response, signify that the cached
1353 version on the client is still valid (depending on the client's HTTP request). HTTP caching
1354 semantics should be leveraged as much as possible to support the efficiency and scalability of
1355 this HTTP-based resolution system. The recommended use of HTTP caching headers is
1356 described in more detail in section 3.5.1.

1357 Any ultimate response besides a HTTP 2XX or 304 should be considered an error in the
1358 resolution process. There is no restriction on intermediate redirects (i.e., 3XX result codes) or
1359 other result codes (e.g., a 100 HTTP response) that eventually result in a 2XX or 304 response
1360 through normal operation of [RFC2616]. The content of this ultimate response will be a new XRI
1361 Descriptor for the context of the qualified sub-segment being resolved.

1362 If there are more sub-segments in the XRI authority segment, the process iterates with the next
1363 sub-segment. If there are no more sub-segments, the final context (as described by the final XRI
1364 Descriptor retrieved) can be used for local access services as described in section 3.4.

1365 3.2.5 Examples

1366 Following is an example of resolving the authority portion of this XRI:

```
1367 xri:=example.home.base/foo.bar
```

1368 Assume that the URI for the "=" global context symbol is "http://equals.example.org/xri-resolve"
1369 (found in **XRIDescriptor/XRIAuthority/URI** of the XRI Descriptor for this community). As
1370 explained in 3.2.3, this information, which provides a starting point for resolution, is known *a priori*
1371 and is part of the configuration of the resolver.

1372

1373



1373 **Resolving “=example”**

1374 The following HTTP request is made to “equals.example.org”:

```
1375 GET /xri-resolve/.example HTTP/1.1
1376 If-Modified-Since: Fri, 31 Oct 2003 19:43:31 GMT
1377
1378 <other HTTP headers>
```

1379

1380 The following HTTP response is received from “equals.example.org” (the content has changed
1381 since “Fri, 31 Oct 2003 19:43:31 GMT”):

```
1382 200 OK HTTP/1.1
1383 Content-Type: application/xrid+xml
1384 Expires: Fri, 7 Nov 2003 19:43:31 GMT
1385 <other HTTP headers>
1386
1387 <XRIDescriptor xmlns="...">
1388 <Resolved>.example</Resolved>
1389 <XRIAuthority>
1390 <URI>
1391 http://xri.example.com/xri-resolve/
1392 </URI>
1393 </XRIAuthority>
1394 <LocalAccess>...</LocalAccess>
1395
1396 </XRIDescriptor>
```

1397

1398 **Resolving “=example.home”**

1399 Appending the next [qualified](#) sub-segment “_home” to the URI “http://xri.example.com/xri-resolve/”
1400 yields the URI “http://xri.example.com/xri-resolve/.home”, and the following HTTP request is
1401 made to xri.example.com:

```
1402 GET /xri-resolve/.home HTTP/1.1
1403 <other HTTP headers>
```

1404

1405 The following HTTP response is received from xri.example.com:

```
1406 200 OK HTTP/1.1
1407 Content-Type: application/xrid+xml
1408 If-Modified-Since: Fri, 31 Oct 2003 19:43:32 GMT
1409 <other HTTP headers>
1410
1411 <XRIDescriptor xmlns="...">
1412 <Resolved>.home</Resolved>
1413 <XRIAuthority>
1414 <URI>
1415 http://xri.example.com/xri-resolve/.home/
1416 </URI>
1417 </XRIAuthority>
1418 <LocalAccess>...</LocalAccess>
1419 ...
1420 </XRIDescriptor>
```

1421

1422 **Resolving “=example.home.base”**

1423 Appending the next [qualified](#) sub-segment “_base” to the URI
1424 “http://xri.example.com/xri-resolve/[.home/](#)” gives the URI
1425 “http://xri.example.com/xri-resolve/[.home/](#).base”:
1426

```

1427 GET /xri-resolve/.home/.base HTTP/1.1
1428 If-Modified-Since: Fri, 31 Oct 2003 19:43:32 GMT
1429
1430 <other HTTP headers>

```

1431

1432 The following HTTP response is received from xri.example.com:

```

1433 200 OK HTTP/1.1
1434 Content-type: application/xrid+xml
1435 Expires: Fri, 7 Nov 2003 19:43:33 GMT
1436
1437 <other HTTP headers>
1438
1439 <XRIDescriptor xmlns=""...">
1440 <Resolved>.base</Resolved>
1441 <LocalAccess>
1442 <URI>
1443 http://xri.example.com/xri-local/base/
1444 </URI>
1445 <URI>
1446 https://xri.example.com/xri-local/base/
1447 </URI>
1448 </LocalAccess>
1449 ...
1450 </XRIDescriptor>

```

1451

1452 The result of the final XRI authority resolution step is the set of HTTP and HTTPS URIs shown in
1453 the “LocalAccess” element above that can be used for local access services.



1454 3.2.6 Resolving Cross-References in XRI Authorities

1455 A sub-segment within an XRI authority segment may be a cross-reference (see sections 2.1.1.4
1456 and 2.1.2). Resolving a cross-reference is identical to resolving any other sub-segment because,
1457 from the standpoint of generic XRI resolution, the cross-reference is considered opaque. In other
1458 words, the value of the cross-reference (including the parentheses) is the literal value of the sub-
1459 segment for the purpose of constructing the Next Resolution URI as described in section 3.2.4.

1460 An exception to the above is a cross-reference that begins with the GCS symbol for annotations
1461 (“!”). Such a cross-reference and the delimiter that precedes it MUST be ignored entirely during
1462 resolution.

1463 [Table 7](#) provides several examples. In each of these examples, sub-segment “b” resolves
1464 to an XRI Authority URI of “http://example.com/xri-authority/”.

1465

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)

1466 Table 7: Examples of the Next Authority URIs constructed using different types of cross-references.

1467 Note that specific identifier communities may specify special resolution rules for specific types of
1468 cross-references, but such extensions are out of scope for this specification.

1469 **3.2.7 User Relative XRIs**

1470 A special case of XRI authority resolution is the user-relative context symbol ("**"). This symbol
1471 means the XRI authority is defined by the user of the XRI rather than specified in the XRI itself.
1472 For example, frequent XRI users could use the "**" symbol to enter their own XRI "shortcuts" or
1473 "speed names" which are resolved by the user's preferred XRI authority. These XRIs are not
1474 resolvable without the *a priori* mapping of the "**" symbol to the base XRI specified by the user.

1475 An XRI beginning with the user-relative context symbol MUST be transformed into an absolute
1476 XRI that does not begin with a user-relative context symbol before it can be resolved using the
1477 resolution protocol defined in this specification. To perform this transformation, the XRI value
1478 following the "**" symbol MUST be treated as a relative XRI reference and resolved relative to a
1479 base XRI defined in section 2.3. The mapping of the "**" symbol to this base XRI is
1480 implementation-dependent; however, the configuration of such mapping SHOULD be easily
1481 available to the user.

1482 Note that in most cases, mapping requires simply replacing the "**" character with a prefix
1483 corresponding to a pre-configured base XRI. For example, if the pre-configured user-relative base
1484 XRI is "@employer/Mary", then the XRI "xri:*workstation" would be converted into
1485 "xri:@employer/Mary/workstation".

1486 **3.3 URI Authority Resolution**

1487 A URI-authority segment (section 2.1.1.1) includes either a DNS name or an IP address that
1488 specifies the location of the endpoint with which to perform local access. This simplifies the
1489 process for converting XRIs with URI authorities into local access URIs. First, the XRI must be
1490 converted into URI-escaped form (section 2.2.4.3). Then the scheme is converted from "xri:" to
1491 "http:", and an HTTP request is performed on the resulting URI, as described in section 3.4,
1492 "Local Access", below.

1493 For example, the XRI "xri://www.example.com/foo.bar" is transformed to the HTTP URI
1494 "http://www.example.com/foo.bar".

1495 The use of URI authorities provides backwards compatibility with the large installed base of DNS-
1496 and IP-identifiable resources. However because URI authorities do not support the additional
1497 layer of abstraction and extensibility represented by XRI authority syntax, URI authorities are not
1498 recommended for new deployments of XRI identifiers.

1499 **3.4 Local Access**

1500 Local access is the process of interacting with a network endpoint to retrieve a representation of a
1501 network resource identified by an XRI.

1502 **3.4.1 Local Access Service Types**

1503 Any number of protocols may be used for local access. This specification defines an
1504 HTTP/HTTPS local access protocol. An LDAP or DSML local access protocol could be defined by
1505 specifying the appropriate transformation of the XRI local part into an LDAP distinguished name
1506 (including normalization of the XRI local path to the LDAP distinguished name syntax.)

1507 Work on such protocols is left to future specifications. To accommodate such work, this
1508 specification reserves a namespace, "\$r.a", for enumerating local access service types. One
1509 enumeration, "X2R", is defined in section 3.2.2 under "/XRIDescriptor/LocalAccess/Service".

1510 **3.4.2 HTTP/HTTPS Local Access**

1511 The HTTP/HTTPS local access protocol does not specify the semantics of the local access
1512 interaction, nor the form of the local access requests. The only semantics defined are those in
1513 **[RFC2616]**. Special attention should be paid to the semantics of the four main HTTP verbs: GET,

1514 PUT, POST, and DELETE. For example, clients performing local access typically would use GET
1515 when wishing to retrieve representations of a resource on the network.

1516 This specification does not impose particular semantics beyond what is defined in **[RFC2616]**, but
1517 users of this specification are encouraged to review the **[REST]** architecture when building
1518 applications using XRIs. Local access is not limited to the REST model of interaction, however.
1519 For example, HTTP local access could be leveraged for the delivery of SOAP messages over
1520 HTTP POST, or via use of the GET HTTP verb as a generic read-only resolution infrastructure.

1521 The HTTP/HTTPS local access binding defined in this section is flexible enough to be used for a
1522 variety of resources. It makes no assumptions about the type of resource identified by the XRI
1523 being resolved. The resource type must be established through the context in which the XRI was
1524 originally used (e.g. an XML document) or discovered through use of the HTTP local access
1525 protocol (e.g., through the HTTP Content-Type header).

1526 **3.4.3 Constructing a Local Access HTTP/HTTPS URI**

1527 This section defines the construction of URIs for local access to resources identified with XRI
1528 authorities. The construction of URIs for local access to resources identified with URI authorities
1529 is defined in section 3.3.

1530 The HTTP/HTTPS URI with which to perform local access is constructed by concatenating the
1531 Local Access URI from the XRI Descriptor (section 3.2.2) with the local part of the XRI.
1532 Specifically, the URI from the element identified with **theby the relative XPATHelement path**
1533 **/XRIDescriptor/LocalAccess/URI in the XRIDescriptor** is concatenated with the URI normal form
1534 (section 2.2.4.3) version of the remaining relative-path (section 2.1). If the LocalAccess URI does
1535 not terminate in a "/", one MUST be inserted before the relative-path.

1536 The following pseudocode describes the process for creating the concrete HTTP/HTTPS URI to
1537 which a local access request is made:

```
1538 concrete-http-uri = localaccess-uri  
1539  
1540 if (localaccess-uri does not end in "/"):  
1541     concrete-http-uri = localaccess-uri + "/"  
1542  
1543 concrete-http-uri = localaccess-uri + uri-escape(relative-path)
```

1544

1545 The verb used in the resulting HTTP/HTTPS request may be any of the verbs defined in
1546 **[RFC2616]**, though not all verbs may be supported at every endpoint. All local access endpoints
1547 SHOULD support at least the GET verb, and this should return either a representation of the
1548 identified resource or metadata about the resource.

1549 The full suite of HTTP content negotiation features is available to clients when performing local
1550 access. For example, if the local access service URI is "http://xri.example.com/xri-local", then the
1551 following local access HTTP request for "xri:=example.home/foo.bar" could be made to
1552 "xri.example.com":

```
1553 GET /xri-local/foo.bar HTTP/1.1  
1554 If-Modified-Since: Fri, 31 Oct 2003 19:43:33 GMT  
1555 <other HTTP headers>  
1556
```

1557

1558 The following HTTP response should then be received from xri.example.com:

```
1559 200 OK HTTP/1.1  
1560 Expires: Sat, 1 Nov 2003 19:43:33 GMT  
1561 Content-Type: text/plain  
1562 <other HTTP headers>  
1563  
1564 This is the result of a local access request.
```

1565 **3.4.4 Using a Cross-Reference to Specify a Representation Type**

1566 A cross-reference MAY be used to specify a desired resource representation type when
1567 performing local access. The namespace “\$r.t” is reserved for this purpose. This specification
1568 does not enumerate such types; they are further defined in the “*XRI Metadata Specification*”
1569 **[XRIMetadata]**.

1570 To specify a particular resource representation type using “\$r.t” metadata, a “\$r.t” cross-reference
1571 is appended to the XRI during a local access request. For example, an RDDL document could be
1572 specified by appending the cross-reference “(\$r.t/RDDL)”.

1573 The following example illustrates this technique. Assuming the original XRI being resolved is
1574 “xri:=example.home/foo.bar” and the local access URI is “http://xri.example.com/xri-local”, the
1575 following HTTP request would request the RDDL document describing this resource:

```
1576 GET /xri-local/foo.bar/%28$r.t%2FRDDL%29 HTTP/1.1  
1577 <other HTTP headers>
```

1578

1579 Note that the cross-reference is escaped per the rules for the URI normal form of an XRI in
1580 section 2.2.4.3.

1581 The resulting HTTP response would be:

```
1582 200 OK HTTP/1.1  
1583 <cache-headers>  
1584 <other HTTP headers>  
1585  
1586 <content of representation of RDDL for xri:=example.home/foo.bar>
```

1587

1588 **3.5 HTTP Headers**

1589 **3.5.1 Caching**

1590 The full caching capabilities of **[RFC2616]** should be leveraged during both identifier authority
1591 resolution and local access. Specifically, implementations of XRI resolution SHOULD implement
1592 the caching model described section 13 of **[RFC2616]**. In particular, the “Expiration Model” of
1593 section 13.2 SHOULD be used, as this requires the fewest round-trip network connections.

1594 All servers providing identifier authority lookup responses SHOULD send the Cache-Control or
1595 Expires headers per section 13.2 of **[RFC2616]**, unless there are overriding security or policy
1596 reasons that dictate otherwise.

1597 **3.5.2 Location**

1598 During identifier authority resolution, “Location” headers may be present per the **[RFC2616]**
1599 specification (i.e., during 3XX redirects). Redirects SHOULD be made cacheable through
1600 appropriate HTTP headers.

1601 During the local access phase, redirects may be returned, and the “Location” field may contain an
1602 HTTP/HTTPS URI or an XRI in URI normal form. This use of redirects constitutes a mapping
1603 facility that allows one XRI to resolve into another during local access. If the local access server is
1604 aware of the HTTP/HTTPS URI where the XRI may be accessed, it can provide a “Location”
1605 header containing an HTTP/HTTPS URI. In this case, it SHOULD provide an “X-XRI-Canonical”
1606 header (see below) to describe the XRI to which the redirection is targeting. If the local access
1607 server knows only of the target XRI, then it MUST return a redirection header (3XX code) with the
1608 “Location” field containing an XRI.

1609 **3.5.3 Content-Location**

1610 “Content-Location” may be used during local access where the resource being accessed is an
1611 “attribute” or “view” of another resource. This usually would occur in the case where metadata is
1612 being accessed using a trailing cross reference to an XRI value under the “\$r.t” namespace (see
1613 section 3.4.4). Such a “Content-Location” header would specify where the resource itself may be
1614 accessible (rather than the metadata). This is not required and MUST NOT be required by
1615 resolving clients for proper operation. The content-location SHOULD be an HTTP/HTTPS URI if
1616 the local access server is aware of the HTTP/HTTPS location, otherwise it MAY be an XRI.

1617 **3.5.4 Content-Type**

1618 “Content-type” is required in the HTTP/HTTPS response during identifier authority resolution,
1619 both when returning an XRI Descriptor and for the HTTP/HTTPS responses during local access.

1620 The “Content-type” header in the 2XX responses in identifier authority resolution for each sub-
1621 segment MUST contain the value “application/xrid+xml”, specifying that the content is an XRI
1622 Descriptor (section 3.2.2).

1623 In local access, clients and servers MAY negotiate content type using standard HTTP content
1624 negotiation features. Whether or not this feature is used, however, the server MUST respond with
1625 an appropriate media type in the “Content-type” header.

1626 **3.5.5 X-XRI-Canonical**

1627 This header is present only in HTTP/HTTPS redirects from local access servers. Its purpose is to
1628 notify a resolving client that the redirect is occurring because the original XRI is a mapping to
1629 another XRI. The value of this header is the target XRI in URI normal form (section 2.2.4.3). This
1630 header MAY be present even when the Location: header is present and contains an XRI. This
1631 header SHOULD be present when the Location: header is present and contains a HTTP/HTTPS
1632 or other URI.

1633 Form:

1634 `X-XRI-Canonical: <xri-in-uri-normal-form>`

1635 **3.6 Other HTTP Features**

1636 HTTP provides a number of other features including transfer-coding, proxying, validation-model
1637 caching, etc. All of these features may be used insofar as they do not conflict with the required
1638 uses of HTTP described in this document.

1639 **3.7 Caching and Efficiency**

1640 Resolution clients are encouraged to perform caching above the HTTP level in addition to at the
1641 HTTP level. For best results, however, resolution clients SHOULD be conservative with caching
1642 expiration semantics, including cache expiration dates. This implies that in a series of HTTP
1643 redirects, for example, the results of the entire process should only be cached as long as the
1644 shortest period of time allowed by any of the intermediate HTTP responses.

1645 Because not all HTTP client libraries expose caching expiration to applications, identifier
1646 authorities and local access servers SHOULD NOT use cacheable redirects with expiration times
1647 which are relatively short compared to the expiration times of other HTTP responses in the
1648 resolution or local access chain. In general, all XRI deployments should be mindful of limitations
1649 in current HTTP clients and proxies.

1650 For XRI Descriptors, the cache expiration time may also be shortened by the expiration time
1651 provided in the XRI Descriptor at **/XRIDescriptor/Expires** (if present). That is, if the expiration
1652 time in **/XRIDescriptor/Expires** is sooner than the expiration time calculated from the HTTP

1653 caching semantics, then the XRI Descriptor SHOULD be discarded before the expiration time in
1654 **/XRIDescriptor/Expires**.

1655 With both application-level and HTTP-level caching, the resolution process is designed to have
1656 minimal overhead. In particular, because each **qualified** sub-segment of an XRI authority is
1657 resolved separately, each step of that resolution is a completely independent, cacheable HTTP
1658 request. For this reason, resolution of top-level (leftmost) **qualified** sub-segments, which are
1659 common to more identifiers, will naturally result in a greater number of cache hits than resolution
1660 of **qualified** sub-segments further to the right.

1661 **3.8 Points of Extensibility**

1662 The XRI resolution scheme described here leverages extensible mechanisms such as HTTP
1663 and XML to provide maximum flexibility. Specifically, changes or additions can be made at the
1664 following points of extensibility:

- 1665 • HTTP negotiation of content types, language, encoding, etc.
- 1666 • Use of HTTP verbs such as POST, PUT and DELETE during local access.
- 1667 • Use of HTTP redirects (3XX) or other response codes during identifier authority
1668 resolution or local access.
- 1669 • Insertion of new elements or attributes in the XRI Descriptor.
- 1670 • Use of cross-references within XRIs, particularly for associating new types of metadata
1671 with a resource (see section 3.4.4 for an example).

1672

1673 4 Security and Data Protection

1674 4.1 Secure Resolution

1675 The resolution protocol described in section 3 is not intrinsically trustworthy. It is expected that, in
1676 practice, some combination of DNSSEC, SSL, TLS, and other existing technologies will be
1677 employed to increase the security of the resolution process.

1678 While such enhancements are outside the scope of this specification, an XRI Secure Resolution
1679 Specification is a future deliverable of the OASIS XRI TC. Additional follow-on work is also
1680 expected to define best practices and facilitate interoperability.

1681 4.2 XRI Metadata

1682 The use of cross-references employing the GCS "\$" symbol for encoding XRI metadata in an XRI
1683 (section 2.2.4.2) may involve other security and data protection considerations that are outside
1684 the scope of this specification. These considerations are addressed in the "XRI Metadata
1685 Specification" [XRIMetadata].

1686 4.3 XRI Usage in Legacy Infrastructure

1687 Where XRIs are used within the Internet and other computing infrastructure, the security and data
1688 protection considerations are similar to those of other URI schemes. In this context the material in
1689 [RFC2396bis], section 7, *Security Considerations*, is particularly informative. It includes a
1690 discussion of the following topics:

- 1691 • Reliability and Consistency
- 1692 • Malicious Construction
- 1693 • Rare IP Address Formats
- 1694 • Sensitive Information
- 1695 • Semantic Attacks

1696 This material notes that "a URI does not in itself pose a direct security threat." In the case of
1697 XRIs, this statement remains true only in legacy environments. As noted below, it may not be true
1698 for new infrastructure that builds on the extensibility of XRI architecture. Such applications must
1699 be developed with independent security reviews for the specific scenarios in which XRIs are
1700 used.

1701 4.4 XRI Usage in Evolving Infrastructure

1702 As XRIs are adopted as abstract identifiers, it is anticipated that new services will be developed
1703 that take advantage of their extensibility. In particular, XRIs may enable new solutions to security
1704 and data protection problems that are not possible using existing URI schemes.

1705 For example, XRI cross-reference syntax permits the inclusion of identifier metadata such as an
1706 encrypted or integrity-checked path, query, or fragment. Cross-references can also be used to
1707 indicate methods of obfuscating, proxying, or redirecting resolution to prevent the exposure of
1708 private or sensitive data. These capabilities may enable new security and data protection features
1709 at the fundamental level of resource identifiers.

1710 A complete discussion of this topic is beyond the scope of this document. However, as a
1711 consequence of XRI extensibility, it is not possible to make definitive statements regarding
1712 security and data protection considerations relating to XRIs.

5 References

5.1 Normative

- 1715 [RFC1737] K. Sollins, L. Masinter, *Functional Requirements for Uniform Resource Names*,
1716 <http://www.ietf.org/rfc/rfc1737.txt>, RFC 1737, December 1994.
- 1717 [RFC2046] N. Borenstein, N. Freed, *Multipurpose Internet Mail Extensions (MIME) Part Two:*
1718 *Media Types*, <http://www.ietf.org/rfc/rfc2046.txt>, RFC 2046, November 1996.
- 1719 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
1720 <http://www.ietf.org/rfc/rfc2119.txt>, RFC 2119, March 1997.
- 1721 [RFC2141] R. Moats, *URN Syntax*, <http://www.ietf.org/rfc/rfc2141.txt>, IETF RFC 2141, May
1722 1997.
- 1723 [RFC2234] D. H. Crocker and P. Overell, *Augmented BNF for Syntax Specifications: ABNF*,
1724 <http://www.ietf.org/rfc/rfc2234.txt>, RFC 2234, November 1997.
- 1725 [RFC2396] T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifiers (URI):*
1726 *Generic Syntax*, <http://www.ietf.org/rfc/rfc2396.txt>, RFC 2396, August 1998.
- 1727 [RFC2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee,
1728 *Hypertext Transfer Protocol -- HTTP/1.1*, <http://www.ietf.org/rfc/rfc2616.txt>, RFC 2616, June
1729 1999.
- 1730 [RFC2718] L. Masinter, H. Alvestrand, D. Zigmond, R. Petke, *Guidelines for New URL*
1731 *Schemes*, <http://www.ietf.org/rfc/rfc2718.txt>, RFC 2718, November 1999.
- 1732 [RFC2732] R. Hinden, B. Carpenter, L. Masinter, *Format for Literal IPv6 Addresses in URL's*,
1733 <http://www.ietf.org/rfc/rfc2732.txt>, RFC 2732, December, 1999.
- 1734 [RFC3066] H. Alvestrand, *Tags for the Identification of Languages*,
1735 <http://www.ietf.org/rfc/rfc3066.txt>, RFC 3066, January, 2001.
- 1736 [RFC3305] M. Mealing, R. Denenberg, *Uniform Resource Identifiers (URIs), URLs, and*
1737 *Uniform Resource Names (URNs): Clarifications and Recommendations*,
1738 <http://www.ietf.org/rfc/rfc3305.txt>, RFC 3305, August 2002.
- 1739 [RFC3490] P. Faltstrom, P. Hoffman, A. Costello, *Internationalizing Domain Names in*
1740 *Applications (IDNA)*, <http://www.ietf.org/rfc/rfc3490>, RFC 3490, March 2003.
- 1741 [RFC3491] P. Hoffman, M. Blanchet, *Nameprep: A Stringprep Profile for Internationalized*
1742 *Domain Names (IDN)*, <http://www.ietf.org/rfc/rfc3491>, RFC 3491, March 2003.
- 1743 [UML] Object Management Group, *Unified Modeling Language (UML) Version 1.5*,
1744 <http://www.omg.org/technology/documents/formal/uml.htm>, March 1, 2003.
- 1745 [Unicode] The Unicode Consortium. The Unicode Standard, Version 4.0.0, defined by: *The*
1746 *Unicode Standard, Version 4.0* (Boston, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1)
- 1747 [UniXML] Duerst, M. and A. Freytag, *Unicode in XML and other Markup Languages*,
1748 Unicode Technical Report #20, World Wide Web Consortium Note, February 2002.
- 1749 [UTR15] M. Davis, M. Duerst, *Unicode Normalization Forms*,
1750 <http://www.unicode.org/unicode/reports/tr15/tr15-23.html>, April 17, 2003.
- 1751 [XML] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, *Extensible Markup Language*
1752 *(XML) 1.0 (Second Edition) W3C Recommendation*, <http://www.w3.org/TR/REC-xml>, October
1753 2000.
- 1754 [XMLSchema2] P. Biron, A. Malhotra, *XML Schema Part 2: Datatypes W3C*
1755 *Recommendation*, <http://www.w3.org/TR/xmlschema-2/>, May 2001.

1756 **5.2 Informative**

- 1757 **[IRI]** M. Duerst, M. Suignard, *Internationalized Resource Identifiers (IRIs)*,
1758 <http://www.ietf.org/internet-drafts/draft-duerst-iri-05.txt>, Work-In-Progress, October 2003.
- 1759 **[REST]** <http://internet.conveyor.com/RESTwiki/moin.cgi/FrontPage>
- 1760 **[RFC2396bis]** R. Fielding, *Uniform Resource Identifiers (URI): Generic Syntax*, Internet Draft
1761 draft-fielding-uri-rfc2396bis-03, <http://www.apache.org/~fielding/uri/rev-2002/rfc2396bis.html>,
1762 Work-In-Progress, June 2003.
- 1763 **[XRIMetadata]** OASIS XRI Technical Committee, *Extensible Resource Identifier (XRI)*
1764 *Metadata Specification*, <http://www.oasis-open.org/committees/xri/xri-metadata-1.0>, Work-In-
1765 Progress, January 2003.
- 1766 **[XRIReqs]** G. Wachob, D. Reed, M. Le Maitre, D. McAlpin, D. McPherson, *Extensible*
1767 *Resource Identifier (XRI) Requirements and Glossary v1.0*, [http://www.oasis-](http://www.oasis-open.org/apps/org/workgroup/xri/download.php/2523/xri-requirements-and-glossary-v1.0.doc)
1768 [open.org/apps/org/workgroup/xri/download.php/2523/xri-requirements-and-glossary-v1.0.doc](http://www.oasis-open.org/apps/org/workgroup/xri/download.php/2523/xri-requirements-and-glossary-v1.0.doc),
1769 June 2003.
- 1770

1771

Appendix A. Collected ABNF for XRI (Normative)

1772 This section contains the complete ABNF for XRI syntax, which includes the complete ABNF for
 1773 URI from **[RFC2396bis]** of which XRI syntax is a superset. XRI productions use green shading,
 1774 and productions inherited from URI use yellow shading. A valid XRI MUST conform to this ABNF.

1775

1776

```
abs-path      = "/" path-segments
```

1777

1778

```
absolute-xri = "xri:" global-path
```

1779

1780

```
alphanum     = ALPHA / DIGIT
```

1781

1782

```
authority    = [ userinfo "@" ] host [ ":" port ]
```

1783

1784

```
authority-path = URI-authority / XRI-authority
```

1785

1786

```
dec-octet   = DIGIT                               ; 0-9
              / %x31-39 DIGIT                     ; 10-99
              / "1" 2DIGIT                         ; 100-199
              / "2" %x30-34 DIGIT                  ; 200-249
              / "25" %x30-35                       ; 250-255
```

1792

```
delims      = "<" / ">" / "%" / DQUOTE
```

1793

1794

```
escaped     = "%" HEXDIG HEXDIG
```

1795

1796

```
excluded    = invisible / delims / unwise
```

1797

1798

```
fragment    = *( pchar / "/" / "?" )
```

1799

1800

```
gcs-char    = "+" / "=" / "@" / "$" / "*" / "!"
```

1801

1802

```
global-path = authority-path [ local-path ]
```

1803

1804

```
h4          = 1*4HEXDIG
```

1805

1806

```
hier-part   = net-path / abs-path / rel-path
```

1807

1808

```
host        = [ hostname / IPv4address / IPv6reference ]
```

1809

1810

```
hostname    = idomainlabel qualified
```

1811

1812

```
idomainlabel = ( ALPHA / ucschar ) *( alphanum / ucschar / "-" )
```

1813

1814

```
invisible   = CTL / SP / %x80-FF
```

1815

1816

```
IPv4address = dec-octet "." dec-octet "." dec-octet "." dec-octet
```

1817

1818

```
IPv6address = 6( h4 ":" ) ls32
              / "::" 5( h4 ":" )                               ls32
              / [ h4 ] "::" 4( h4 ":" )                       ls32
              / [ *1( h4 ":" ) h4 ] "::" 3( h4 ":" )         ls32
              / [ *2( h4 ":" ) h4 ] "::" 2( h4 ":" )         ls32
              / [ *3( h4 ":" ) h4 ] "::" h4 ":"              ls32
              / [ *4( h4 ":" ) h4 ] "::"                      ls32
              / [ *5( h4 ":" ) h4 ] "::"                      h4
              / [ *6( h4 ":" ) h4 ] "::"
```

1827

```

1828
1829 IPv6reference = "[" IPv6address "]"
1830
1831 local-path = "/" relative-path
1832
1833 ls32 = ( h4 ":" h4 ) / IPv4address
1834 ; least-significant 32 bits of address
1835
1836 mark = "-" / "_" / "." / "!" / "~" / "*" / "'" / "(" / ")"
1837
1838 net-path = "/" authority [ abs-path ]
1839
1840 path-segments = segment *( "/" segment )
1841
1842 pchar = unreserved / escaped / ";" /
1843 ":" / "@" / "&" / "=" / "+" / "$" / ","
1844
1845 port = *DIGIT
1846
1847 qualified = *( "." idomainlabel ) [ "." ]
1848
1849 query = *( pchar / "/" / "?" )
1850
1851 relative-path = [ xri-segments ] [ "?" xri-query ]
1852 [ "#" xri-fragment ]
1853
1854 relative-xri = local-path / relative-path
1855
1856 rel-path = path-segments
1857
1858 reserved = "/" / "?" / "#" / "[" / "]" / ";" /
1859 ":" / "@" / "&" / "=" / "+" / "$" / ","
1860
1861 scheme = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )
1862
1863 segment = *pchar
1864
1865 sub-segment = 1*xri-pchar / xref
1866
1867 ucschar = %xA0-D7FF / %xF900-FDCF / %xFDF0-FFEF /
1868 %x10000-1FFFD / %x20000-2FFFD / %x30000-3FFFF /
1869 %x40000-4FFFF / %x50000-5FFFF / %x60000-6FFFF /
1870 %x70000-7FFFF / %x80000-8FFFF / %x90000-9FFFF /
1871 %xA0000-AFFFF / %xB0000-BFFFF / %xC0000-CFFFF /
1872 %xD0000-DFFFF / %xE1000-EFFFF
1873
1874 unreserved = ALPHA / DIGIT / mark
1875
1876 unwise = "{" / "}" / "|" / "\" / "^" / "`"
1877
1878 URI = scheme ":" hier-part [ "?" query ] [ "#" fragment ]
1879
1880 URI-authority = "/" [ userinfo "@" ] host [ ":" port ]
1881
1882 uric = reserved / unreserved / escaped
1883
1884 userinfo = *( unreserved / escaped / ";" /
1885 ":" / "&" / "=" / "+" / "$" / "," )
1886
1887 xref = "(" ( xri-value / URI ) ")"
1888
1889 xref-authority = xref ( "." sub-segment / ":" sub-segment )
1890 *( "." sub-segment / ":" sub-segment )

```

```

1891
1892 XRI = absolute-xri / relative-xri
1893
1894 XRI-authority = ( gcs-char [ xri-segment ] ) / xref-authority
1895
1896 xri-characters = xri-reserved / xri-unreserved / escaped
1897
1898 xri-fragment = [ xref ] * ( xri-pchar / "." / ":" / "/" / "?" )
1899
1900 xri-mark = "-" / "_" / "~" / "'"
1901
1902 xri-pchar = xri-unreserved / escaped / ";" / "!" / "*"
1903 "@" / "&" / "=" / "+" / "$" / ","
1904
1905 xri-query = [ xref ] * ( xri-pchar / "." / ":" / "/" / "?" )
1906
1907 xri-reserved = "/" / "?" / "#" / "[" / "]" / "(" / ")" / ";" / ":" /
1908 ", " / "." / "&" / "@" / "=" / "+" / "*" / "$" / "!"
1909
1910 xri-segment-val = xri-segment / "." / ".."
1911
1912 xri-segment = ( [ "." ] sub-segment / ":" sub-segment )
1913 *( "." sub-segment / ":" sub-segment )
1914
1915 xri-segments = xri-segment-val *( "/" [ xri-segment-val ] )
1916
1917 xri-unreserved = ALPHA / DIGIT / ucschar / xri-mark
1918
1919 xri-value = global-path / local-path / relative-path
1920
1921

```

1922

Appendix B. XML Schema for XRI Descriptor (Normative)

1923

```
1924 <xs:schema targetNamespace="xri:$r.s/XRIDescriptor" xmlns:xs="http://www.w3.org/2001/XMLSchema"
1925 xmlns="xri:$r.s/XRIDescriptor" elementFormDefault="qualified" attributeFormDefault="unqualified">
1926   <xs:complexType name="XRIDescriptorType">
1927     <xs:sequence>
1928       <xs:element name="Resolved" type="ResolvedType" minOccurs="0" maxOccurs="unbounded"/>
1929       <xs:element name="Expires" type="ExpiresType" minOccurs="0"/>
1930       <xs:element name="XRIAuthority" type="XRIAuthorityType" minOccurs="0"/>
1931       <xs:element name="LocalAccess" type="LocalAccessType" minOccurs="0" maxOccurs="unbounded"/>
1932       <xs:element name="Mapping" type="MappingType" minOccurs="0" maxOccurs="unbounded"/>
1933       <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
1934     </xs:sequence>
1935     <xs:anyAttribute namespace="##other" processContents="lax"/>
1936   </xs:complexType>
1937   <xs:complexType name="XRIAuthorityType">
1938     <xs:sequence>
1939       <xs:element name="URI" type="URIType" maxOccurs="unbounded"/>
1940       <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
1941     </xs:sequence>
1942     <xs:anyAttribute namespace="##other" processContents="lax"/>
1943   </xs:complexType>
1944   <xs:complexType name="LocalAccessType">
1945     <xs:sequence>
1946       <xs:element name="Service" type="ServiceType" minOccurs="0"/>
1947       <xs:element name="Type" type="TypeType" minOccurs="0" maxOccurs="unbounded"/>
1948       <xs:element name="URI" type="URIType" maxOccurs="unbounded"/>
1949       <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
1950     </xs:sequence>
1951     <xs:anyAttribute namespace="##other" processContents="lax"/>
1952   </xs:complexType>
1953   <xs:element name="XRIDescriptor" type="XRIDescriptorType"/>
1954   <xs:complexType name="ResolvedType">
1955     <xs:simpleContent>
1956       <xs:extension base="xs:string">
1957         <xs:anyAttribute namespace="##other" processContents="lax"/>
1958       </xs:extension>
1959     </xs:simpleContent>
1960   </xs:complexType>
1961   <xs:complexType name="URIType">
1962     <xs:simpleContent>
1963       <xs:extension base="xs:anyURI">
1964         <xs:anyAttribute namespace="##other" processContents="lax"/>
1965       </xs:extension>
1966     </xs:simpleContent>
1967   </xs:complexType>
1968   <xs:complexType name="ExpiresType">
1969     <xs:simpleContent>
1970       <xs:extension base="xs:dateTime">
1971         <xs:anyAttribute namespace="##other" processContents="lax"/>
1972       </xs:extension>
1973     </xs:simpleContent>
1974   </xs:complexType>
1975   <xs:complexType name="ServiceType">
1976     <xs:simpleContent>
1977       <xs:extension base="xs:anyURI">
1978         <xs:anyAttribute namespace="##other" processContents="lax"/>
1979       </xs:extension>
1980     </xs:simpleContent>
1981   </xs:complexType>
```



```
1982 <xs:complexType name="TypeType">
1983 <xs:simpleContent>
1984 <xs:extension base="xs:string">
1985 <xs:anyAttribute namespace="##other" processContents="lax"/>
1986 </xs:extension>
1987 </xs:simpleContent>
1988 </xs:complexType>
1989 <xs:complexType name="MappingType">
1990 <xs:simpleContent>
1991 <xs:extension base="xs:string">
1992 <xs:anyAttribute namespace="##other" processContents="lax"/>
1993 </xs:extension>
1994 </xs:simpleContent>
1995 </xs:complexType>
1996 </xs:schema>
1997
```

Appendix C. Transforming HTTP URIs to XRI (Non-Normative)

1998
1999

2000 To leverage existing infrastructure, it may sometimes be useful to convert HTTP URIs into XRIs.
2001 Because XRI syntax is, for the most part, a superset of generic URI syntax, the majority of HTTP
2002 URIs can be converted to valid XRIs simply by replacing the scheme “http” with “xri”. Special
2003 consideration, however, must be given to HTTP URIs employing the characters in the “xri-
2004 reserved” production of this specification that differ from those in the “reserved” production of
2005 **[RFC2396]** (as amended by **[RFC2732]**). These include opening parenthesis (“(“), closing
2006 parenthesis (“)”), dot (“.”), asterisk (“*”), and exclamation point (“!”).

2007 Typically, characters in the “reserved” production of **[RFC2396]** that appear in an HTTP URI as
2008 normal characters (i.e. not as syntactic delimiters) are escaped encoded. However, this is not
2009 required in all cases. **[RFC2396]** says

2010 “Characters in the ‘reserved’ set are not reserved in all contexts. The set of characters
2011 actually reserved within any given URI component is defined by that component. In general, a
2012 character is reserved if the semantics of the URI changes if the character is replaced with its
2013 escaped US-ASCII encoding.”

2014 Characters in the “xri-reserved” set that are properly left un-escaped in an HTTP URI may be
2015 semantically significant when the HTTP URI is converted to an XRI. For example,

2016 `http://www.example.com/example1:example2`

2017 is a valid HTTP URI even though it contains an unescaped reserved character – a colon (“:”) –
2018 because section 3.3 of **[RFC2396]** explicitly omits this character from the reserved set for “path”
2019 components. The same unescaped character in an XRI, however, will be interpreted as a
2020 delimiter. If the colon character should not be understood as a delimiter in the resulting XRI, it
2021 must be escaped during conversion. The same applies to the other characters mentioned above.

2022 Generally, any character not in the “xri-pchar” set that appears in the “abs_path”, “query”, or
2023 “fragment” components of the HTTP URI will need to be escaped when converting to an XRI. This
2024 avoids misinterpretation in the resulting XRI following the guidance in section 2.2.4 of this
2025 specification.

2026 Exceptions are possible. For example, if the author of the above HTTP URI intended the colon
2027 character to be interpreted as described in this specification, or if its use would not be
2028 misinterpreted, then it may be left in its unescaped form.

2029 In addition, it may be beneficial to escape other characters like the percent (“%”) character,
2030 particularly if it may be necessary to convert the resulting XRI back to an HTTP URI. Whether
2031 such additional escaping is desirable or not depends on the intended use of the resulting XRI, the
2032 context in which it will appear, how it is intended to be resolved, etc.

2033 It is worth noting that some rare forms of HTTP URIs can result in XRIs that are misleading to the
2034 reader. For example, the following unusual HTTP URI is valid per **[RFC2396]**.

2035 `http://@example.com/example1`

2036 When converted to an XRI, as

2037 `xri://@example.com/example1`

2038 a casual reader could easily misinterpret the “uri-authority” component as an “xri-authority”.
2039 Similarly, a URI with an authority segment like

2040 `http://=bob@example.com/example1`

2041 could be similarly misinterpreted.

2042

2043

Appendix D. Acknowledgments

2044 The editors would also like to thank the following people who participated in the XRI TC and/or
2045 provided input and review of this specification (affiliations listed for OASIS members):
2046

2047 Thomas Bikeev (EAN International), Winston Bumpus (formerly of Novell), James Bryce Clark
2048 (OASIS), Matthey Dovey (Individual), Lars Marius Garshol, Steve Green (Epok), Lance Hood
2049 (Epok), Phillipe LeBlanc (GemPlus), Marc LeMaitre (OneName), Rajeev Maria (Visa
2050 International), Adarbad Master (Epok), John McGarvey (IBM), Davis McPherson (Epok), Mike
2051 Mealling (Verisign), Reva Modi (Infosys), Joseph Moeller (EDS), Brian Nimmo (Epok), Mary
2052 Nishikawa (Individual), Eamonn Neylon (Individual), Masaki Nishitani (NRI), Norman Paskin,
2053 Krishnan Rajagopalan (Novell), Chetan Sabnis (Epok), Jim Schreckengast (formerly of Gemplus),
2054 Tomonori Seki (NRI), Xavier Serret (Gemplus), Terence Spielman (Visa International), Marc
2055 Stephenson (TSO), Geoffrey Strongin (AMD), Bernard Vatant, John Veizades (Visa
2056 International), Bill Washburn (XNSORG), Tetsu Watanabe (NRI), Dave Wentker (Visa
2057 International), Loren West (Epok), and Michael Willett (Wave Systems).

2058

2059 A special acknowledgement to Jerry Kindall (Epok) for a full editorial review.

2060

2061 Also, the authors of and contributors to the following documents and specifications are
2062 acknowledged for the intellectual foundations of the XRI specification:

- 2063
- RFC 1737
- 2064
- RFC 2396 (and RFC 2396bis)
- 2065
- RFC 2616
- 2066
- RFC 2718
- 2067
- RFC 3401-3405 (DDDS)
- 2068
- REST Architecture
- 2069
- IRI – Internationalized Resource Identifiers draft
- 2070
- XNS
- 2071

2072

Appendix E. Revision History

Rev	Date	By Whom	What
1.0-cd	2003-01-09	All Editors	Initial document.

2073



2074

Appendix F. Notices

2075 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
2076 that might be claimed to pertain to the implementation or use of the technology described in this
2077 document or the extent to which any license under such rights might or might not be available;
2078 neither does it represent that it has made any effort to identify any such rights. Information on
2079 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
2080 website. Copies of claims of rights made available for publication and any assurances of licenses
2081 to be made available, or the result of an attempt made to obtain a general license or permission
2082 for the use of such proprietary rights by implementors or users of this specification, can be
2083 obtained from the OASIS Executive Director.

2084 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
2085 applications, or other proprietary rights which may cover technology that may be required to
2086 implement this specification. Please address the information to the OASIS Executive Director.

2087 **Copyright © OASIS Open 2004. All Rights Reserved.**

2088 This document and translations of it may be copied and furnished to others, and derivative works
2089 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
2090 published and distributed, in whole or in part, without restriction of any kind, provided that the
2091 above copyright notice and this paragraph are included on all such copies and derivative works.
2092 However, this document itself does not be modified in any way, such as by removing the
2093 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
2094 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
2095 Property Rights document must be followed, or as required to translate it into languages other
2096 than English.

2097 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
2098 successors or assigns.

2099 This document and the information contained herein is provided on an "AS IS" basis and OASIS
2100 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
2101 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
2102 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
2103 PARTICULAR PURPOSE.