



Extensible Resource Identifier (XRI) Syntax V2.0

Committee Draft 01, 14 March 2005

Document identifier:

xri-syntax-V2.0-cd-01

Location:

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

Editors:

Drummond Reed, Cordance <drummond.reed@cordance.net>
Dave McAlpin, Epok <dave.mcalpin@epok.net>

Contributors:

Peter Davis, Neustar <peter.davis@neustar.biz>
Nat Sakimura, NRI <n-sakimura@nri.co.jp>
Mike Lindelsee, Visa International <mlindels@visa.com>
Gabe Wachob, Visa International <gwachob@visa.com>

Abstract:

This document is the normative technical specification for XRI generic syntax. For a non-normative introduction to the uses and features of XRIs, see *Introduction to XRIs* at **[XRIntro]**. For the HTTP-based XRI resolution protocol, see *Extensible Resource Identifier (XRI) Resolution V2.0* at **[XRResolution]**. For the set of XRIs defined to provide metadata about other XRIs, see *Extensible Resource Identifier (XRI) Metadata V2.0* at **[XRIMetadata]**.

Status:

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

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

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

The non-normative errata page for this specification is located at <http://www.oasis-open.org/committees/xri>.

39 Table of Contents

40	Introduction	4
41	1.1 Overview of XRIs.....	4
42	1.1.1 Generic Syntax	4
43	1.1.2 URI, URL, URN, and XRI.....	5
44	1.2 Terminology and Notation	6
45	1.2.1 Keywords	6
46	1.2.2 Syntax Notation.....	6
47	2 Syntax	7
48	2.1 Characters.....	7
49	2.1.1 Character Encoding	7
50	2.1.2 Reserved Characters	7
51	2.1.3 Unreserved Characters.....	7
52	2.1.4 Percent-Encoded Characters	8
53	2.1.4.1 Encoding XRI Metadata.....	8
54	2.1.5 Excluded Characters.....	8
55	2.2 Syntax Components.....	9
56	2.2.1 Authority	9
57	2.2.1.1 XRI Authority.....	10
58	2.2.1.2 Global Context Symbol (GCS) Authority	10
59	2.2.1.3 IRI Authority	10
60	2.2.2 Cross-References	11
61	2.2.3 Path.....	11
62	2.2.4 Query	12
63	2.2.5 Fragment.....	12
64	2.3 Transformations	13
65	2.3.1 Transforming XRI References into IRI and URI References.....	13
66	2.3.2 Escaping Rules for XRI Syntax.....	13
67	2.3.3 Transforming IRI References into XRI References	15
68	2.4 Relative XRI References	15
69	2.4.1 Reference Resolution	15
70	2.4.2 Reference Resolution Examples	16
71	2.4.2.1 Normal Examples	16
72	2.4.2.2 Abnormal Examples.....	16
73	2.4.3 Leading Segments Containing a Colon	17
74	2.4.4 Leading Segments Beginning with a Cross-Reference	17
75	2.5 Normalization and Comparison.....	17
76	2.5.1 Case.....	17
77	2.5.2 Encoding, Percent-Encoding, and Transformations	18
78	2.5.3 Optional Syntax.....	18
79	2.5.4 Cross-References	18
80	2.5.5 Canonicalization.....	18
81	3 Security and Data Protection Considerations	20

82	3.1 Cross-References	20
83	3.2 XRI Metadata	20
84	3.3 Spoofing and Homographic Attacks.....	20
85	3.4 UTF-8 Attacks	20
86	3.5 XRI Usage in Evolving Infrastructure	21
87	4 References.....	22
88	4.1 Normative	22
89	4.2 Informative.....	23
90	Appendix A. Collected ABNF for XRI (Normative)	24
91	Appendix B. Transforming HTTP IRIs to XRIs (Non-Normative)	27
92	Appendix C. Glossary	28
93	Appendix D. Acknowledgments.....	33
94	Appendix E. Notices	34
95		

96 Introduction

97 1.1 Overview of XRIs

98 Extensible Resource Identifiers (XRIs) provide a standard means of abstractly identifying a
99 resource independent of any particular concrete representation of that resource—or, in the case
100 of a completely abstract resource, of any representation at all.

101 As shown in Figure 1, XRIs build on the foundation established by URIs (Uniform Resource
102 Identifiers) and IRIs (Internationalized Resource Identifiers) as defined by **[URI]** and **[IRI]**,
103 respectively.



104
105

Figure 1: The relationship of XRIs, IRIs, and URIs

106 The IRI specification created a new identifier by extending the unreserved character set to include
107 characters beyond those allowed in generic URIs. It also defined rules for transforming this
108 identifier into a syntactically legal URI. Similarly, this specification creates a new identifier, an
109 XRI, that extends the syntactic elements (but not the character set) allowed in IRIs. To
110 accommodate applications that expect IRIs or URIs, this specification also defines rules for
111 transforming an XRI reference into a valid IRI or URI reference.

112 Although an XRI is not a Uniform Resource Name (URN) as defined in *URN Syntax* **[RFC2141]**,
113 an XRI consisting entirely of persistent segments is designed to meet the requirements set out in
114 *Functional Requirements for Uniform Resource Names* **[RFC1737]**.

115 This document specifies the normative syntax for XRIs, along with associated normalization,
116 processing and equivalence rules. Two additional specifications complete the XRI 2.0 suite:

- 117 • *XRI Resolution* **[XRIResolution]** specifies both a standard and a trusted HTTP-based
118 resolution protocol for XRIs. Use of these protocols is not required; XRIs may also be
119 resolved using other protocols or resolution mechanisms.
- 120 • *XRI Metadata* **[XRIMetadata]** specifies a small set of standard metadata identifiers registered
121 under the XRI global context symbol "\$" that may be used to describe the contents of an XRI
122 reference.

123 See also *An Introduction to XRIs* **[XRIIntro]** for a non-normative introduction to XRI 2.0 syntax,
124 resolution, and metadata via a set of practical examples.

125 1.1.1 Generic Syntax

126 XRI syntax follows the same basic pattern as IRI and URI syntax. A fully-qualified XRI consists of
127 the prefix "xri://" followed by the same four components as a generic authority-based IRI or URI.

128

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

129 The definitions of these components are, for the most part, supersets of the equivalent
130 components in the generic IRI or URI syntax. One advantage of this approach is that the vast
131 majority of HTTP URIs and IRIs, which derive directly from generic URI syntax, can be
132 transformed to valid XRIs simply by changing the scheme from “http” to “xri”. This transformation
133 is discussed in Appendix B, “Transforming HTTP IRIs to XRIs”.

134 XRI syntax extends generic IRI syntax in the following four ways:

- 135 1. *Persistent and reassignable segments.* Unlike generic URI syntax, XRI syntax allows the
136 internal components of an XRI reference to be explicitly designated as either persistent or
137 reassignable.
- 138 2. *Cross-references.* Cross-references allow XRI references to contain other XRI references
139 or IRIs as syntactically-delimited sub-segments. This provides syntactic support for
140 “compound identifiers”, i.e., the use of well-known, fully-qualified identifiers within the
141 context of another XRI reference. Typical uses of cross-references include using well-
142 known types of metadata in an XRI reference (such as versioning metadata as defined in
143 the *XRI Metadata* specification [**XRIMetadata**]), or the use of globally-defined identifiers
144 to mark parts of an XRI reference as having application- or vocabulary-specific
145 semantics.
- 146 3. *Global context symbols.* While XRI syntax supports the same generic syntax used in IRIs
147 for DNS and IP authorities, it also provides shorthand symbols for establishing the
148 abstract global context of an identifier.
- 149 4. *Standardized federation.* Federated identifiers are those delegated across multiple
150 authorities, such as DNS names. Generic URI syntax leaves the syntax for federated
151 identifiers up to individual URI schemes, with the exception of explicit support for IP
152 addresses. XRI syntax standardizes federation of both persistent and reassignable
153 identifiers at any level of the path.

154 1.1.2 URI, URL, URN, and XRI

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

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

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

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

168 This conclusion is shared by [**URI**] which states in section 1.1.3:

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

173 XRIs are consistent with this philosophy. Although XRIs are designed to fulfill the requirements of
174 abstract “names” that are resolved into concrete locators, the XRI syntax does not distinguish
175 between identifiers that represent “names”, “locators” or “characteristics.”

176 1.2 Terminology and Notation

177 1.2.1 Keywords

178 The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”,
179 “SHOULD NOT”, “RECOMMENDED”, “MAY” and “OPTIONAL” in this document are to be
180 interpreted as described in [RFC2119]. When these words are not capitalized in this document,
181 they are meant in their natural language sense.

182 1.2.2 Syntax Notation

183 This specification uses the syntax notation employed in [IRI]: Augmented Backus-Naur Form
184 (ABNF), defined in [RFC2234]. Although the ABNF defines syntax in terms of the US-ASCII
185 character encoding, XRI syntax should be interpreted in terms of the character that the ASCII-
186 encoded octet represents, rather than the octet encoding itself, as explained in [URI]. As with
187 URIs, the precise bit-and-byte representation of an XRI reference on the wire or in a document is
188 dependent upon the character encoding of the protocol used to transport it, or the character set of
189 the document that contains it.

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

193 To simplify comparison between generic XRI syntax and generic IRI syntax, the ABNF
194 productions that are unique to XRIs are shown with light green shading, while those inherited
195 from [IRI] are shown with light yellow shading.

196 | This is an example of ABNF specific to XRI.

197 ; This is an example of ABNF inherited from IRI.

198 Lastly, because the prefix “xri://” is optional in absolute XRIs that use a global context symbol
199 (see section 2.2.1.2), some example XRIs are shown without this prefix.

200 2 Syntax

201 This section defines the normative syntax for XRIs. Note that additional constraints are inherited
202 from **[IRI]** and **[URI]**, as defined in section 2.2. Also note that some productions in the XRI ABNF
203 are ambiguous. As with IRIs and URIs, a “first-match-wins” rule is used to disambiguate
204 ambiguous productions. See **[URI]** for more details.

205 2.1 Characters

206 XRI character set and encoding are inherited from **[IRI]**, which is a superset of generic URI
207 syntax as defined in **[URI]**.

208 2.1.1 Character Encoding

209 The standard character encoding of XRI is UTF-8, as recommended by **[RFC2718]**. When an XRI
210 reference is presented as a human-readable identifier, the representation of the XRI reference in
211 the underlying document may use the character encoding of the underlying document. However,
212 this representation must be converted to UTF-8 before the XRI can be processed outside the
213 document.

214 2.1.2 Reserved Characters

215 The overall XRI reserved character set is the same as the reserved character set defined by
216 **[URI]** and **[IRI]**. Due to the extended syntax of XRIs, however, the allocation of reserved
217 characters between the “general delimiters” and “sub-delimiters” productions is different. Those
218 characters that have defined semantics in generic XRI syntax appear in the xri-gen-delims
219 production. Those characters that do not have defined semantics but that are reserved for use as
220 implementation-specific delimiters appear in the xri-sub-delims production. The rgcs-char
221 production that appears in xri-gen-delims below is discussed in section 2.2.1.2.

```
222 xri-reserved      = xri-gen-delims / xri-sub-delims
223 xri-gen-delims   = ":" / "/" / "?" / "#" / "[" / "]" / "(" / ")"
224                 / "*" / "!" / rgcs-char
225 xri-sub-delims   = "&" / ";" / "," / "'"
```

226 If an XRI reserved character is used as a data character and not as a delimiter, the character
227 MUST be percent-encoded per the rules in section 2.1.4, “Percent-Encoded Characters”. XRI
228 references that differ in the percent-encoding of a reserved character are not equivalent.

229 2.1.3 Unreserved Characters

230 The characters allowed in XRI references that are not reserved are called unreserved. XRI has
231 the same set of unreserved characters as the "iunreserved" production in **[IRI]**.

```
232 iunreserved      = ALPHA / DIGIT / "-" / "." / "_" / "~" / uchar
233 uchar            = %xA0-D7FF / %xF900-FDCF / %xFDF0-FFEF
234                 / %x10000-1FFFD / %x20000-2FFFD / %x30000-3FFFD
235                 / %x40000-4FFFD / %x50000-5FFFD / %x60000-6FFFD
236                 / %x70000-7FFFD / %x80000-8FFFD / %x90000-9FFFD
237                 / %xA0000-AFFFD / %xB0000-BFFFD / %xC0000-CFFFD
238                 / %xD0000-DFFFD / %xE1000-EFFFD
```

239 Percent-encoding unreserved characters in an XRI does not change what resource is identified
240 by that XRI. However, it may change the result of an XRI comparison (see section 2.5,
241 "Normalization and Comparison"), so unreserved characters SHOULD NOT be percent-encoded.

242 2.1.4 Percent-Encoded Characters

243 XRIs follow the same rules for percent-encoding as IRIs and URIs. That is, any *data* character in
244 an XRI reference MUST be percent-encoded if it does not have a representation using an
245 unreserved character but SHOULD NOT be percent-encoded if it does have a representation
246 using an unreserved character. Delimiters in an XRI reference that have a representation using a
247 reserved character MUST NOT be percent-encoded.

248 An XRI reference thus percent-encoded is said to be in *XRI-normal form*. Not all XRI references
249 in XRI-normal form are syntactically legal IRI or URI references. Rules for converting an XRI
250 reference to a valid IRI or URI reference are discussed in section 2.3.1. An XRI reference is in
251 XRI-normal form if it is minimally percent-encoded and matches the ABNF provided in this
252 document, but it is a valid IRI or URI reference only after it is percent-encoded according to the
253 transformation described in section 2.3.1.

254 A percent-encoded octet is a character triplet consisting of the percent character "%" followed by
255 the two hexadecimal digits representing that octet's numeric value.

256 `escaped = "%" HEXDIG HEXDIG`

257 The uppercase hexadecimal digits "A" through "F" are equivalent to the lowercase digits "a"
258 through "f", respectively. XRI references that differ only in the case of hexadecimal digits used in
259 percent-encoded octets are equivalent. For consistency, XRI generators and normalizers
260 SHOULD use uppercase hexadecimal digits for percent-encoded triplets.

261 Note that a % symbol used to represent itself in an XRI reference (i.e., as data and not to
262 introduce a percent-encoded triplet) must be percent-encoded.

263 2.1.4.1 Encoding XRI Metadata

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

269 XRI syntax offers an option for encoding this language metadata using a cross-reference
270 beginning with the GCS "\$" symbol (see section 2.2.1.2). The top level authority for \$I language
271 metadata is the *XRI Metadata Specification [XRIMetadata]*, specifically section 2. See also
272 section 3 for "\$d" date/time metadata, section 4 for "\$v" version metadata, and section 5 for "\$-"
273 annotation metadata.

274 2.1.5 Excluded Characters

275 Certain characters, such as "space", are excluded from the XRI syntax and must be percent-
276 encoded in order to be represented within an XRI. Systems responsible for accepting or
277 presenting XRI references may choose to percent-encode excluded characters on input and/or
278 decode them prior to display, as described in section 2.1.4. A string that contains these
279 characters in a non-percent-encoded form, however, is not a valid XRI.

280 Note that presenting "space" or other whitespace characters in a non-percent-encoded form is not
281 recommended for several reasons. First, it is often difficult to visually determine the number of
282 spaces or other characters composing a block of whitespace, leading to transcription errors.
283 Second, the space character is often used to delimit an XRI reference, so non-percent-encoded
284 whitespace characters can make it difficult or impossible to determine where the identifier ends.
285 Finally, non-percent-encoded whitespace can be used to maliciously construct subtly different

286 identifiers intended to mislead the reader. For these reasons, non-percent-encoded whitespace
287 characters SHOULD be avoided in presentation, and alternatives to whitespace as a logical
288 separator within XRIs (such as dots or hyphens) SHOULD be used whenever possible.

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

291 “The UCS contains many areas of characters for which there are strong visual
292 look-alikes. Because of the likelihood of transcription errors, these also should be
293 avoided. This includes the full-width equivalents of Latin characters, half-width
294 Katakana characters for Japanese, and many others. This also includes many
295 look-alikes of ‘space’, ‘delims’, and ‘unwise’, characters excluded in **[RFC3491]**.”

296 “Additional information is available from **[UniXML]**. **[UniXML]** is written in the
297 context of running text rather than in the context of identifiers. Nevertheless, it
298 discusses many of the categories of characters not appropriate for IRIs.”

299 2.2 Syntax Components

300 XRI syntax builds on generic IRI (and ultimately, URI) syntax. However because XRI syntax
301 includes syntactic elements other than those defined in **[IRI]** and **[URI]**, this specification defines
302 a new protocol element, "XRI", along with rules for transforming XRI references into generic IRI or
303 URI references for applications that expect them (see section 2.3.1, “Transforming XRI
304 References into IRI and URI References”). An XRI reference MUST be constructed such that it
305 qualifies as a valid IRI as defined by **[IRI]** when converted to IRI-normal form and such that it
306 qualifies as a valid URI as defined by **[URI]** when converted to URI-normal form.

307 As with URIs, an XRI must be in absolute form, while an XRI reference may be either an XRI or a
308 relative XRI reference.

```
309 XRI-reference = XRI / relative-XRI-ref
310 XRI           = [ "xri://" ] xri-hier-part [ "?" iquery ]
311              [ "#" ifragment ]
312 absolute-XRI = [ "xri://" ] xri-hier-part [ "?" iquery ]
313 xri-value    = xri-no-scheme / relative-XRI-ref
314 xri-no-scheme = xri-hier-part [ "?" iquery ]
315              [ "#" ifragment ]
316 relative-XRI-ref = xri-path [ "?" iquery ] [ "#" ifragment ]
317 xri-hier-part   = ( xri-authority / iauthority )
318              [ xri-path-absolute ] / ipath-empty
```

319 An XRI begins with an optional prefix “xri://” followed by the same set of hierarchical components
320 as a URI – authority, path, query, and fragment. An XRI is always in absolute form. A relative XRI
321 reference consists of an XRI path followed by an optional XRI query and optional XRI fragment.
322 The absolute-XRI production is provided for contexts that require an XRI in absolute form but that
323 do not allow the fragment identifier.

324 Finally, in certain contexts where XRIs are used exclusively, the prefix “xri://” is redundant. These
325 contexts can use the xri-value production, which includes all levels of XRI paths.

326 2.2.1 Authority

327 XRIs support the same types of authorities as generic IRIs, called *IRI authorities*. XRIs also
328 support an additional type of abstract identification authority called an *XRI authority*.

329 **2.2.1.1 XRI Authority**

330 There are two ways to express an XRI authority: using a global context symbol (GCS), or using a
331 cross-reference (abbreviated in the ABNF as *xref*). Cross-references are covered in section 2.2.2.

332 `xri-authority = gcs-authority / xref-authority`

333 **2.2.1.2 Global Context Symbol (GCS) Authority**

334 XRIs offer a simple, compact syntax for indicating the logical global context of an identifier: a
335 single prefix character called a *global context symbol*.

336 `gcs-authority = pgcs-authority / rgcs-authority`

337 `pgcs-authority = "!" xri-subseg-pt-nz *xri-subseg`

338 `rgcs-authority = rgcs-char xri-segment`

339 `rgcs-char = "=" / "@" / "+" / "$"`

340 The global context symbol characters were selected from the set of symbol characters that are
341 valid in a URI under **[URI]**. The bang character, “!”, which is used uniformly in XRI syntax to
342 indicate a persistent identifier segment, serves as the GCS character for global persistent
343 identifiers. The other GCS characters may be used to indicate the global context of either a
344 persistent or a reassignable identifier as shown in Table 1 below:
345

Symbol Character	Authority Type	Establishes Global Context For
=	Person	Identifiers for whom the authority is an individual person.
@	Organization	Identifiers for whom the authority is an organization or a resource in an organizational context.
+	General public	Identifiers for whom the authority is the general public, i.e., that represent generic “dictionary” concepts for which there is no specific authority. (In the English language, for example, these would be the generic nouns.)
\$	Standards body	Identifiers for whom the authority is a specification from a standards body, for example, other XRI specifications (such as <i>XRI Resolution</i> [XRIResolution] and <i>XRI Metadata</i> [XRIMetadata]), other OASIS specifications, or (using cross-references) other standards bodies.

346 Table 1: XRI global context symbols.

347 **2.2.1.3 IRI Authority**

348 XRIs support the same type of authority defined by the “iauthority” production of **[IRI]**.

349 `iauthority = [iuserinfo "@"] ihost [":" port]`

350 `iuserinfo = *(iunreserved / pct-encoded / sub-delims / ":")`

351 `ihost = IP-literal / IPv4address / ireg-name`

352 `port = *DIGIT`

353 The syntax is inherited directly from **[IRI]**. First, the “userinfo” sub-component permits the
354 identification of a user in the context of a host. Next, the “ihost” sub-component has three options
355 for identifying the host: a registered name (such as a domain name), an IPv4 address, or an IPv6
356 literal.

357 A host identifier can be followed by an optional port number. The XRI syntax specification does
358 not define a default port because it is expected this will be inherited from the resolution protocol,
359 such as the HTTP/HTTPS protocol specified in **[XRIResolution]**. Therefore, if the port is omitted
360 in an XRI, it is undefined.

361 Note that authority segments that begin with GCS characters or cross-references (see below)
362 may match both the “iauthority” and the “xri-authority” productions. For instance, “!1”,
363 “@example”, “=example”, “+example”, “\$example” and “(=example)” all match both productions.
364 As with all XRI syntax, the “first-match-wins” rule is used to resolve ambiguities. Consequently, all
365 the examples listed above would be considered XRI authorities, not IRI authorities.

366 2.2.2 Cross-References

367 Cross-references are the primary extensibility mechanism in XRI. They allow an identifier
368 assigned in one context to be reused in another context, permitting identifiers to be shared across
369 contexts. This simplifies identifying logically equivalent resources across hierarchies (a directory
370 concept referred to as “polyarchy”).

371 A cross-reference is syntactically delimited by enclosing it in parentheses, similar to the way an
372 IPv6 literal is encapsulated in square brackets as specified in **[RFC2732]**. A cross-reference may
373 contain either an XRI reference or an absolute IRI.

```
374 xref = "( ( XRI-reference / IRI ) )"
```

375 It is important that the value of a cross-reference be syntactically unambiguous, whether it is an
376 absolute IRI or one of the various forms of an XRI reference. Therefore special attention must be
377 paid to relative XRI references to avoid ambiguity, as discussed in section 2.4.3.

378 A cross-reference may appear at any node of any XRI except within an IRI authority segment. A
379 cross-reference as the very first sub-segment in an XRI is a valid top-level XRI authority.

```
380 xref-authority = xref *xri-subseg
```

381 This syntax allows any globally-unique identifier in any URI scheme (e.g., an HTTP URI, mailto
382 URI, URN etc.) to specify a global XRI authority.

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

385 2.2.3 Path

386 As with IRIs, the XRI path component is a hierarchal sequence of path segments separated by
387 slash (“/”) characters and terminated by the first question-mark (“?”) or number sign (“#”) character,
388 or by the end of the XRI reference. But while an IRI path segment is considered
389 opaque by a generic URI processor, an XRI path segment can be parsed by an XRI processor
390 into two types of sub-segments: ** segments* (pronounced “star segments”) and *! segments*
391 (pronounced “bang segments”).

```
392 xri-path = xri-path-absolute  
393 / xri-path-noscheme  
394 / ipath-empty  
395 xri-path-absolute = "/" [ xri-segment-nz *( "/" xri-segment ) ]
```

```

396 xri-path-noscheme = xri-subseg-od-nx *xri-subseg-nc
397                   *( "/" xri-segment )
398
398 xri-segment       = xri-subseg-od *xri-subseg
399
399 xri-segment-nz    = xri-subseg-od-nz *xri-subseg
400
400 xri-subseg        = ( "*" / "!" ) (xref / *xri-pchar)
401
401 xri-subseg-nc     = ( "*" / "!" ) (xref / *xri-pchar-nc)
402
402 xri-subseg-od     = [ "*" / "!" ] (xref / *xri-pchar)
403
403 xri-subseg-od-nz  = [ "*" / "!" ] (xref / 1*xri-pchar)
404
404 xri-subseg-od-nx  = [ "*" / "!" ] 1*xri-pchar-nc
405
405 xri-subseg-pt-nz  = "!" (xref / 1*xri-pchar)

```

406 * segments are used to specify *reassignable identifiers*—identifiers that may be reassigned by an
 407 identifier authority to represent a different resource at some future date. ! segments are used to
 408 specify *persistent identifiers*—identifiers that are permanently assigned to a resource and will not
 409 be reassigned at a future date. The default is a * segment, so no leading star (“*”) is required for
 410 the first (or only) sub-segment.

411 An XRI path segment may contain the same characters as a URI path segment plus the
 412 expanded UCS character set inherited from [IRI]. If a star (“*”) or bang (“!”) appears in a path of
 413 an XRI reference, it will be interpreted as a sub-segment delimiter. If this interpretation is not
 414 desired for these characters, or for any other special XRI delimiters, these characters MUST be
 415 percent-encoded when they appear in the path segment. See section 2.1.4, “Percent-Encoded
 416 Characters”.

```

417 xri-pchar          = iunreserved / pct-encoded / xri-sub-delims / ":"
418
418 xri-pchar-nc       = iunreserved / pct-encoded / xri-sub-delims

```

419 With the exception of star (“*”), bang (“!”) and cross-reference delimiters, an XRI path segment is
 420 considered opaque by generic XRI syntax. As with IRIs, XRI extensions or generating
 421 applications may define special meanings for other XRI reserved characters for the purpose of
 422 delimiting extension-specific or generator-specific sub-components.

423 Note that XRI syntax is slightly more restrictive than URI syntax in that the first segment of an
 424 absolute XRI path may never be empty, even in the absolute form of an XRI.

425 2.2.4 Query

426 The XRI query component is identical to the IRI query component as described in section 2.2 of
 427 [IRI].

```

428 iquery            = *( ipchar / iprivate / "/" / "?" )

```

429 2.2.5 Fragment

430 XRI syntax also supports fragments as described in section 2.2 of [IRI].

```

431 ifragment         = *( ipchar / "/" / "?" )

```

432 Since XRI federation syntax can inherently address attributes or sub-resources to any depth,
433 fragments are supported primarily for compatibility with generic URI syntax. XRIs can also employ
434 cross-references to identify media types or other alternative representations of a resource. See
435 section 2.2.2

436 2.3 Transformations

437 2.3.1 Transforming XRI References into IRI and URI References

438 Although XRIs are intended to be used by applications that understand them natively, it may also
439 be desirable to use them in contexts that do not recognize an XRI reference but that allow an
440 Internationalized Resource Identifier reference as described in [IRI], or a fully-conformant URI
441 reference as defined by [URI].

442 This section specifies the steps for transforming an XRI reference into a valid IRI reference. At
443 the completion of these steps, the XRI reference is in *IRI-Normal Form*. An XRI reference in IRI-
444 Normal Form may then be mapped into a valid URI reference by following the algorithms defined
445 in section 3.1 of [IRI]. After that mapping, the XRI reference is in *URI-Normal Form*.

446 Applications MUST transform XRI references to IRI references using the following steps (or a
447 process that achieves exactly the same result). Before applying these steps, the XRI reference
448 must be in XRI-normal form as defined in section 2.1.4.

- 449 1. If the XRI reference is not encoded in UTF-8, convert the XRI reference to a sequence of
450 characters encoded in UTF-8, normalized according to Normalization Form C (NFC) as
451 defined in [UTR15].
- 452 2. If the XRI reference is not relative (i.e., if it matches the “XRI” ABNF production) and the
453 optional “xri:” prefix has been omitted, prepend “xri:” to the XRI reference.
- 454 3. Optionally add XRI metadata using cross-references as defined in section 2.1.4.1. Note
455 that the addition of XRI metadata may change the resulting IRI or URI reference for the
456 purposes of comparison. The significance or insignificance of specific types of XRI
457 metadata is specified in *Extensible Resource Identifier (XRI) Metadata V2.0*
458 [XRIMetadata].
- 459 4. Apply the XRI escaping rules defined in section 2.3.2. Note that this step is not
460 idempotent (i.e., it may yield a different result if applied more than once), so it is very
461 important that implementers not apply this step more than once to avoid changing the
462 semantics of the identifier.

463 At the completion of step 4, the percent-encoded XRI reference may be used as an IRI reference.
464 An XRI reference in this form is said to be in *IRI-normal form*.

465 Applying this conversion does not change the equivalence of the identifier, with the possible
466 exception of the addition of XRI metadata as discussed in Step 3.

467 In general, an application SHOULD use the least-transformed version appropriate for the context
468 in which the identifier appears. For example, if the context allows an XRI reference directly, the
469 identifier SHOULD be an XRI reference in XRI-normal form as described in section 2.1.4. If the
470 context allows an IRI reference but not an XRI reference, the identifier SHOULD be in IRI-normal
471 form. Only when context allows neither XRI nor IRI references should URI-normal form be used.

472 2.3.2 Escaping Rules for XRI Syntax

473 This section defines rules for preventing misinterpretation of XRI syntax when an XRI reference is
474 evaluated by a non-XRI-aware parser.

475 The first rule deals with cross-references as explained in section 2.2.2. Since a cross-reference
476 contains either an IRI or an XRI reference (which itself may contain further nested IRIs or XRI
477 references), it may include characters that, if not escaped, would cause misinterpretation when

478 the XRI reference is used in a context that expects an IRI or URI reference. Consider the
479 following XRI:

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

481 The generic parsing algorithm described in **[URI]** would separate the above XRI into the following
482 components:

```
483 scheme = xri  
484 authority = @example  
485 path = /(xri://@example2/abc  
486 query = id=1)
```

487 The desired separation is:

```
488 scheme = xri  
489 authority = @example  
490 path = /(xri://@example2/abc?id=1)  
491 query = <undefined>
```

492 To avoid this type of misinterpretation, certain characters in a cross-reference must be percent-
493 encoded when transforming an XRI reference into IRI-normal form. In particular, the question
494 mark (“?”) character must be percent-encoded as “%3F” and the number sign “#” character must
495 be percent-encoded as “%28”.

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

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

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

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

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

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

504 instead of the intended:

```
505 xri://@example.com/
```

506 This is because the “merge” algorithm in section 5.2.3 of **[URI]** is defined in terms of the last
507 (right-most) slash character. This problem is avoided by encoding slashes within cross-references
508 as “%2F”. Following this rule, the above example would be expressed as:

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

510 Ambiguity is also possible if an XRI reference in XRI-normal form contains characters that have
511 been percent-encoded to indicate that they should not be interpreted as delimiters. For example,
512 consider the following XRI in XRI-normal form:

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

514 This slash character between “c” and “d” is percent-encoded to show that it’s not a syntactical
515 element of the XRI, i.e., that it should be interpreted as data and not as a delimiter. To preserve

516 this type of distinction when converting an XRI reference to an IRI reference, the percent “%”
517 character must be percent-encoded as “%25”. Following this rule, the above example fully
518 converted would be:

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

520 To summarize, the following four special rules **MUST** be applied during step 4 of section 2.3.1.
521 Before applying these rules, the XRI reference **MUST** be in XRI-normal form and all IRIs in cross-
522 references **MUST** be in a percent-encoded form appropriate to their schemes.

- 523 1. Percent-encode all percent “%” characters as “%25” across the entire XRI reference.
- 524 2. Percent-encode all number sign “#” characters that appear within a cross-reference as
525 “%23”.
- 526 3. Percent-encode all question mark “?” characters that appear within a cross-reference as
527 “%3F”.
- 528 4. Percent-encode all slash “/” characters that appear within a cross-reference as “%2F”.

529 **2.3.3 Transforming IRI References into XRI References**

530 Transformation of an XRI reference in IRI-normal form into an XRI reference in XRI-normal form
531 **MUST** use the following steps (or a process that achieves the same result).

- 532 1. If the XRI reference is not encoded in UTF-8, convert the XRI reference to a sequence of
533 characters encoded in UTF-8, normalized according to Normalization Form C (NFC) as
534 defined in **[UTR15]**.
- 535 2. Perform the following special conversions for XRI syntax:
 - 536 a. Convert all percent-encoded slash (“/”) characters to their corresponding octets.
 - 537 b. Convert all percent-encoded question mark (“?”) characters to their
538 corresponding octets.
 - 539 c. Convert all percent-encoded number sign (“#”) characters to their corresponding
540 octets.
 - 541 d. Convert all percent-encoded percent (“%”) characters to their corresponding
542 octets.

543 Note that this process is not idempotent (i.e., it may yield a different result if applied more than
544 once), so it is very important that implementers only apply this process to XRI references in IRI-
545 normal form. If it is applied to an XRI reference in XRI-normal form, the resulting identifier may
546 not be equivalent to the XRI reference before transformation.

547 **2.4 Relative XRI References**

548 **2.4.1 Reference Resolution**

549 For XRI references in IRI-normal form or URI-normal form, resolving a relative XRI reference into
550 an absolute XRI reference is straightforward. If the base XRI and the relative XRI reference are in
551 IRI-normal form, section 6.5 of **[IRI]** applies. If the base XRI and the relative XRI reference are in
552 URI-normal form, section 5 of **[URI]** applies.

553 It is important that XRI references appear in a form appropriate to their context (i.e., in URI-
554 normal form in contexts that expect URI references and in IRI-normal form in contexts that expect
555 IRI references), since the algorithms described in **[IRI]** and **[URI]** may produce incorrect results
556 when applied to XRI references in XRI-normal form, particularly when those XRI references
557 contain cross-references.

558 In contexts that allow a native XRI reference (i.e., an XRI reference in XRI-normal form), it may
559 be useful to perform relative reference resolution without first converting to IRI- or URI-normal
560 form. In fact, it may be difficult or impossible to convert to IRI- or URI-normal form without first
561 resolving the relative XRI reference to an absolute XRI. The algorithms described in section 5 of
562 **[URI]** apply to XRI references in XRI-normal form provided that the processor:

- 563 • treats the characters allowed in IRI references but not in URI references the same as it
564 treats unreserved characters in URI references (as required by section 5 of [IRI]) and
- 565 • treats all characters within all cross-references the same as unreserved characters in URI
566 references (i.e., treats cross-references as opaque with respect to relative reference
567 resolution).

568 2.4.2 Reference Resolution Examples

569 The following are examples of relative XRI reference resolution. These examples are very similar
570 to the examples for resolving relative references in [URI]. Starting with the following base XRI in
571 XRI-normal form:

```
572 xri://@a*a/!b!b/c*c/(xri://@d*d/e)?q
```

573 a relative reference is transformed to its target XRI as shown in the following examples.

574 2.4.2.1 Normal Examples

```
575 !g!g = xri://@a*a/!b!b/c*c/!g!g
576 ./!g!g = xri://@a*a/!b!b/c*c/!g!g
577 !g!g/ = xri://@a*a/!b!b/c*c/!g!g/
578 /!g!g = xri://@a*a/!g!g
579 //!g!g = Not a legal relative XRI reference
580 ?y = xri://@a*a/!b!b/c*c/(xri://@d*d/e)?y
581 !g!g?y = xri://@a*a/!b!b/c*c/!g!g?y
582 #s = xri://@a*a/!b!b/c*c/(xri://@d*d/e)?q#s
583 !g!g#s = xri://@a*a/!b!b/c*c/!g!g#s
584 !g!g?y#s = xri://@a*a/!b!b/c*c/!g!g?y#s
585 ;x = xri://@a*a/!b!b/c*c/;x
586 !g!g;x = xri://@a*a/!b!b/c*c/!g!g;x
587 !g!g;x?y#s = xri://@a*a/!b!b/c*c/!g!g;x?y#s
588 = xri://@a*a/!b!b/c*c/(xri://@d*d/e)?q
589 . = xri://@a*a/!b!b/c*c/
590 ./ = xri://@a*a/!b!b/c*c/
591 .. = xri://@a*a/!b!b/
592 ../ = xri://@a*a/!b!b/
593 ../!g!g = xri://@a*a/!b!b/!g!g
594 ../.. = xri://@a*a/
595 ../.. / = xri://@a*a/
596 ../.. /!g!g = xri://@a*a/!g!g
```

597 2.4.2.2 Abnormal Examples

598 As in IRIs and URIs, the "." syntax cannot be used to change the authority component of an XRI.

```
599 ../... /!g!g = xri://@a*a/!g!g
600 ../... /.. /!g!g = xri://@a*a/!g!g
```

601 As in IRIs and URIs, "." and ".." have a special meaning only when they appear as complete path
602 segments.

```
603 ./!g!g = xri://@a*a/!g!g
604 ../!g!g = xri://@a*a/!g!g
605 !g!g. = xri://@a*a/!b!b/c*c/!g!g.
606 .!g!g = xri://@a*a/!b!b/c*c/.!g!g
607 !g!g.. = xri://@a*a/!b!b/c*c/!g!g..
608 ..!g!g = xri://@a*a/!b!b/c*c/..!g!g
```

609 XRI parsers, like IRI and URI parsers, must be prepared for superfluous or nonsensical uses of
610 "." and "..".


```

611      ./../!g!g      = xri://@a*a/!b!b/!g!g
612      ./!g!g/.      = xri://@a*a/!b!b/c*c/!g!g/
613      !g!g/./h      = xri://@a*a/!b!b/c*c/!g!g/h
614      !g!g/./h      = xri://@a*a/!b!b/c*c/h
615      !g!g;x=1/./y   = xri://@a*a/!b!b/c*c/!g!g;x=1/y
616      !g!g;x=1/./y   = xri://@a*a/!b!b/c*c/y

```

617 XRI parsers, like IRI and URI parsers, must take care to separate the reference's query and/or
618 fragment components from the path component before merging it with the base path and
619 removing dot-segments.

```

620      !g!g?y/./x     = xri://@a*a/!b!b/c*c/!g!g?y/./x
621      !g!g?y/./x     = xri://@a*a/!b!b/c*c/!g!g?y/./x
622      !g!g#s/./x     = xri://@a*a/!b!b/c*c/!g!g#s/./x
623      !g!g#s/./x     = xri://@a*a/!b!b/c*c/!g!g#s/./x

```

624 2.4.3 Leading Segments Containing a Colon

625 **[URI]** points out that relative URI references with an initial segment containing a colon may be
626 subject to misinterpretation:

627 "A path segment that contains a colon character (e.g., 'this:that') cannot be used
628 as the first segment of a relative-path reference because it would be mistaken for
629 a scheme name. Such a segment must be preceded by a dot-segment (e.g.,
630 './this:that') to make a relative-path reference."

631 Relative XRI references can be similarly misinterpreted. If any segment prior to the first slash ("/")
632 character in a relative XRI reference contains a colon, the relative XRI reference must be
633 rewritten to begin either with "*", if appropriate, or "/.". Thus, "a:b" becomes either "*a:b" or "/a:b".

634 2.4.4 Leading Segments Beginning with a Cross-Reference

635 A path segment that begins with a cross-reference cannot be used as the first segment of a
636 relative reference because it would be mistaken for an xref-authority. As with a leading segment
637 containing a colon, such a segment must be preceded with "/." to make a relative XRI reference.

638 2.5 Normalization and Comparison

639 In general, the normalization and comparison rules for generic IRIs and URIs specified in Section
640 5 of **[IRI]** and Section 6 of **[URI]** apply to XRIs. This section describes a number of additional XRI-
641 specific rules for normalization and comparison. To reduce the requirements imposed upon a
642 minimally conforming processor, the majority of these rules are RECOMMENDED rather than
643 REQUIRED. An implementation that fails to observe them, however, may frequently treat two
644 XRIs as non-equal when in fact they are equal.

645 Each application that uses XRI references MAY define additional equivalence rules as
646 appropriate. Due to the level of abstraction XRIs provide, such higher-order equivalence rules
647 may be based on indirect comparisons or specified XRI-to-XRI mappings (for example, mappings
648 of reassignable XRIs to persistent XRIs).

649 2.5.1 Case

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

- 651 • Comparison of the scheme component of XRIs and all IRIs used as cross-references is case-
652 insensitive.
- 653 • Comparison of authority components (section 2.2.1) is case-insensitive as defined in **[IRI]**.
- 654 • As specified in section 2.1.4, comparison of characters in a percent-encoding construction is
655 case-insensitive for the hexadecimal digits "A" through "F", i.e. "%ab" is equivalent to "%AB".

656 2.5.2 Encoding, Percent-Encoding, and Transformations

- 657 • Two XRI MUST be considered equivalent if they are character-for-character equivalent.
658 Therefore, they are also equivalent if they are byte-for-byte equivalent and use the same
659 character encoding.
- 660 • Two XRI that differ only in whether unreserved characters are percent-encoded SHOULD be
661 considered equivalent. If one XRI percent-encodes one or more unreserved characters, and
662 another XRI differs only in that the same characters are not percent-encoded, they are
663 equivalent.
- 664 • All forms of an XRI during the transformation process described in section 2.3.1 SHOULD be
665 considered equivalent, assuming the same XRI metadata is inserted as described in section
666 2.3.1.

667 2.5.3 Optional Syntax

- 668 • An “xri-segment” (section 2.2.3) that omits the optional leading star (“*”) SHOULD be
669 considered equivalent to the same “xri-segment” prefixed with an star. For example the
670 segment “/foo*bar” is equivalent to the segment “/*foo*bar”.

671 2.5.4 Cross-References

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

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

- 677 • From the standpoint of XRI syntax, all cross-references beginning with the GCS “\$” symbol
678 SHOULD be considered significant unless stated otherwise in the governing specification, for
679 example *Extensible Resource Identifier (XRI) Metadata V2.0 [XRIMetadata]*. See section
680 2.1.4.1.

681 2.5.5 Canonicalization

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

687 It is nevertheless useful to define guidelines for making XRI references reasonably canonical. XRI
688 references that follow these guidelines will be more consistent in presentation, simpler to process,
689 less prone to false-negative comparisons, and more easily cached. To that end, unless there is a
690 compelling reason to do otherwise, XRI references SHOULD be provided in a form in which:

- 691 • The optional “xri://” prefix is included,
- 692 • The scheme is specified in lowercase,
- 693 • The authority component is specified in lowercase,
- 694 • Percent-encoding uses uppercase A through F,
- 695 • If optional, the leading star in xri-segments is omitted,
- 696 • Unnecessary percent-encoding is not present,
- 697 • `/.` and `/..` are absent in absolute XRI, and
- 698 • Cross-references are reasonably canonical with respect to their schemes.

699 Table 2 illustrates the application of these rules. Although the XRI in the first and second
 700 columns are equivalent, the form in the second column is recommended.
 701

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

702 Table 2: Examples of XRI canonicalization recommendations.

703 3 Security and Data Protection Considerations

704 To a great extent, XRI syntax has the same security considerations as [IRI] and [URI]. In
705 particular the material in [URI], section 7, *Security Considerations*, includes a discussion of the
706 following topics:

- 707 • Reliability and Consistency
- 708 • Malicious Construction
- 709 • Back-End Transcoding
- 710 • Rare IP Address Formats
- 711 • Sensitive Information
- 712 • Semantic Attacks

713 This material notes that “a URI does not in itself pose a direct security threat.” In the case of
714 XRIs, this statement remains true only in legacy environments. As noted below, it may not be true
715 for new infrastructure that builds on the extensibility of XRI architecture. In particular the following
716 features of XRIs deserve special mention.

717 3.1 Cross-References

718 Since cross-references in an XRI can reference other URI schemes, implementation must
719 carefully consider the relevant security considerations for those referenced schemes.

720 3.2 XRI Metadata

721 The use of cross-references employing the GCS “\$” symbol for encoding XRI metadata in an XRI
722 (section 2.1.4.1) may involve other security and data protection considerations that are outside
723 the scope of this specification. These considerations are addressed in *Extensible Resource
724 Identifier (XRI) Metadata V2.0 [XRIMetadata]*.

725 3.3 Spoofing and Homographic Attacks

726 One particularly important security consideration is spoofing, covered first in [URI] and more
727 thoroughly in [IRI] Section 7.5. Spoofing is a semantic attack in which an identifier is deliberately
728 constructed to deceive the user into believing it represents one resource when in fact it
729 represents another. With IRIs in particular, a common example of such an attack is using
730 “homographic” characters (characters from different scripts whose visual appearance is nearly or
731 perfectly identical, e.g., the Latin “A”, the Greek “Alpha”, and the Cyrillic “A”).

732 Spoofing has already been used extensively in email “phishing” attacks. As more browsers add
733 support for Internationalized Domain Names (IDN), it is also beginning to appear in online Web
734 links (“pharming”). Not only are some users less suspicious of URIs on the Web, but the attacker
735 may even obtain a corresponding SSL/TLS certificate for the deceptive URI or IRI to make the
736 fraudulent site look completely secure and legitimate.

737 To help prevent this problem, XRI registries SHOULD institute policies preventing the registration
738 of deceptive XRIs, and user agents that process XRIs SHOULD incorporate safeguards such as
739 warning users when XRIs contain common homographic characters.

740 3.4 UTF-8 Attacks

741 Since XRIs incorporate the use of UTF-8 as specified by [IRI], they can also be subject to UTF-8
742 parsing attacks as described in section 10 of [RFC3629]:

743 “Implementers of UTF-8 need to consider the security aspects of how they
744 handle illegal UTF-8 sequences. It is conceivable that in some circumstances an
745 attacker would be able to exploit an incautious UTF-8 parser by sending it an
746 octet sequence that is not permitted by the UTF-8 syntax.”

747 For more information on these attacks, see section 10 of **[RFC3629]**.

748 **3.5 XRI Usage in Evolving Infrastructure**

749 As XRIs are adopted as abstract identifiers, it is anticipated that new services will be developed
750 that take advantage of their extensibility. In particular, XRIs may enable new solutions to security
751 and data protection problems at the resource identifier level that are not possible using existing
752 URI schemes.

753 For example, XRI cross-reference syntax permits the inclusion of identifier metadata such as an
754 encrypted or integrity-checked path, query or fragment. Cross-references can also be used to
755 indicate methods of obfuscating, proxying or redirecting resolution to prevent the exposure of
756 private or sensitive data.

757 A complete discussion of this topic is beyond the scope of this document. However, as a
758 consequence of XRI extensibility, it is not possible to make definitive statements regarding all
759 security and data protection considerations related to XRIs. New XRI-producing or consuming
760 applications should include independent security reviews for the specific contexts in which they
761 will be used.

4 References

4.1 Normative

- 764 **[IRI]** M. Duerst, M. Suignard, *Internationalized Resource Identifiers (IRIs)*,
765 <http://www.ietf.org/rfc/rfc3987.txt>, RFC 3987, January 2005.
- 766 **[RFC1737]** K. Sollins, L. Masinter, *Functional Requirements for Uniform Resource*
767 *Names*, <http://www.ietf.org/rfc/rfc1737.txt>, RFC 1737, December 1994.
- 768 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
769 <http://www.ietf.org/rfc/rfc2119.txt>, RFC 2119, March 1997.
- 770 **[RFC2141]** R. Moats, *URN Syntax*, <http://www.ietf.org/rfc/rfc2141.txt>, IETF RFC
771 2141, May 1997.
- 772 **[RFC2234]** D. H. Crocker and P. Overell, *Augmented BNF for Syntax Specifications:*
773 *ABNF*, <http://www.ietf.org/rfc/rfc2234.txt>, RFC 2234, November 1997.
- 774 **[RFC2718]** L. Masinter, H. Alvestrand, D. Zigmond, R. Petke, *Guidelines for New*
775 *URL Schemes*, <http://www.ietf.org/rfc/rfc2718.txt>, RFC 2718, November
776 1999.
- 777 **[RFC2732]** R. Hinden, B. Carpenter, L. Masinter, *Format for Literal IPv6 Addresses*
778 *in URL's*, <http://www.ietf.org/rfc/rfc2732.txt>, RFC 2732, December,
779 1999.
- 780 **[RFC3305]** M. Mealing, R. Denenberg, *Uniform Resource Identifiers (URIs), URLs,*
781 *and Uniform Resource Names (URNs): Clarifications and*
782 *Recommendations*, <http://www.ietf.org/rfc/rfc3305.txt>, RFC 3305,
783 August 2002.
- 784 **[RFC3491]** P. Hoffman, M. Blanchet, *Nameprep: A Stringprep Profile for*
785 *Internationalized Domain Names (IDN)*, <http://www.ietf.org/rfc/rfc3491>,
786 RFC 3491, March 2003.
- 787 **[RFC3629]** F. Yergeau, *UTF-8, A Transformation Format of ISO 10646*,
788 <http://www.faqs.org/rfcs/rfc3629.html>, RFC 3629, November, 2003.
- 789 **[UniXML]** Duerst, M. and A. Freytag, *Unicode in XML and other Markup Languages*,
790 Unicode Technical Report #20, World Wide Web Consortium Note,
791 February 2002.
- 792 **[URI]** T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifier*
793 *(URI): Generic Syntax*, <http://www.ietf.org/rfc/rfc3986.txt>, STD 66, RFC
794 3986, January 2005.
- 795 **[UTR15]** M. Davis, M. Duerst, *Unicode Normalization Forms*,
796 <http://www.unicode.org/unicode/reports/tr15/tr15-23.html>, April 17,
797 2003.
- 798 **[XRIMetadata]** D. Reed, *Extensible Resource Identifier (XRI) Metadata V2.0*,
799 <http://docs.oasis-open.org/xri/xri/V2.0/xri-metadata-V2.0-cd-01.pdf>,
800 March 2005.
- 801 **[XRIResolution]** G. Wachob, *Extensible Resource Identifier (XRI) Resolution V2.0*,
802 <http://docs.oasis-open.org/xri/xri/V2.0/xri-resolution-V2.0-cd-01.pdf>,
803 March 2005.

804 **4.2 Informative**

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

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

812

813 Appendix A. Collected ABNF for XRI (Normative)

814 This section contains the complete ABNF for XRI syntax. XRI productions use green shading,
815 while productions inherited from IRI use yellow shading. A valid XRI MUST conform to this ABNF.
816

```
817 XRI           = [ "xri://" ] xri-hier-part [ "?" iquery ]
818               [ "#" ifragment ]

819 xri-hier-part  = ( xri-authority / iauthority ) [ xri-path-absolute ]
820               / ipath-empty

821 XRI-reference  = XRI
822               / relative-XRI-ref

823 absolute-XRI  = [ "xri://" ] xri-hier-part [ "?" iquery ]

824 xri-value      = xri-no-scheme / relative-XRI-ref

825 xri-no-scheme  = xri-hier-part [ "?" iquery ]
826               [ "#" ifragment ]

827 relative-XRI-ref = xri-path [ "?" iquery ] [ "#" ifragment ]

828 xri-authority  = gcs-authority
829               / xref-authority

830 gcs-authority  = pgcs-authority / rgcs-authority

831 pgcs-authority = "!" xri-subseg-pt-nz *xri-subseg

832 rgcs-authority = rgcs-char xri-segment

833 rgcs-char      = "=" / "@" / "+" / "$"

834 xref-authority = xref *xri-subseg

835 xref           = "(" ( XRI-reference / IRI ) ")"

836 xri-path       = xri-path-absolute
837               / xri-path-noscheme
838               / ipath-empty

839 xri-path-absolute = "/" [ xri-segment-nz *( "/" xri-segment ) ]

840 xri-path-noscheme = xri-subseg-od-nx *xri-subseg-nc *( "/" xri-segment )

841 xri-segment     = xri-subseg-od *xri-subseg

842 xri-segment-nz  = xri-subseg-od-nz *xri-subseg

843 xri-subseg      = ( "*" / "!" ) (xref / *xri-pchar)

844 xri-subseg-nc   = ( "*" / "!" ) (xref / *xri-pchar-nc)

845 xri-subseg-od   = [ "*" / "!" ] (xref / *xri-pchar)
```



```

846 xri-subseg-od-nz = [ "*" / "!" ] (xref / 1*xri-pchar)
847 xri-subseg-od-nx = [ "*" / "!" ] 1*xri-pchar-nc
848 xri-subseg-pt-nz = "!" (xref / 1*xri-pchar)
849 xri-pchar = iunreserved / pct-encoded / xri-sub-delims / ":"
850 xri-pchar-nc = iunreserved / pct-encoded / xri-sub-delims
851 xri-reserved = xri-gen-delims / xri-sub-delims
852 xri-gen-delims = ":" / "/" / "?" / "#" / "[" / "]" / "("
853 / ")" / "*" / gcs-char
854 xri-sub-delims = "&" / ";" / "," / "'"

855 IRI = scheme ":" ihier-part [ "?" iquery ]
856 [ "#" ifragment ]
857 scheme = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )
858 ihier-part = "/" iauthority ipath-abempty
859 / ipath-abs
860 / ipath-rootless
861 / ipath-empty
862 iauthority = [ iuserinfo "@" ] ihost [ ":" port ]
863 iuserinfo = *( iunreserved / pct-encoded / sub-delims / ":" )
864 ihost = IP-literal / IPv4address / ireg-name
865 IP-literal = "[" ( IPv6address / IPvFuture ) "]"
866 IPvFuture = "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":" )
867 IPv6address =
868 /
869 / [ h16 ] "::" 5( h16 ":" ) ls32
870 / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
871 / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
872 / [ *3( h16 ":" ) h16 ] "::" h16 ":" ls32
873 / [ *4( h16 ":" ) h16 ] "::" ls32
874 / [ *5( h16 ":" ) h16 ] "::" h16
875 / [ *6( h16 ":" ) h16 ] "::"
876 ls32 = ( h16 ":" h16 ) / IPv4address
877 h16 = 1*4HEXDIG
878 IPv4address = dec-octet "." dec-octet "." dec-octet "." dec-octet
879 dec-octet = DIGIT ; 0-9
880 / %x31-39 DIGIT ; 10-99
881 / "1" 2DIGIT ; 100-199
882 / "2" %x30-34 DIGIT ; 200-249
883 / "25" %x30-35 ; 250-255

```

```

884  ireg-name      = *( iunreserved / pct-encoded / sub-delims )
885  port           = *DIGIT
886  ipath-abempty  = *( "/" isegment )
887  ipath-abs      = "/" [ isegment-nz *( "/" isegment ) ]
888  ipath-rootless = isegment-nz *( "/" isegment )
889  ipath-empty    = 0<ipchar>
890  isegment       = *ipchar
891  isegment-nz    = 1*ipchar
892  iquery         = *( ipchar / iprivate / "/" / "?" )
893  iprivate       = %xE000-F8FF / %xF0000-FFFFD / %x100000-10FFFFD
894  ifragment      = *( ipchar / "/" / "?" )
895  ipchar         = iunreserved / pct-encoded / sub-delims / ":" / "@"
896  iquery         = *( ipchar / iprivate / "/" / "?" )
897  ifragment      = *( ipchar / "/" / "?" )
898  iunreserved    = ALPHA / DIGIT / "-" / "." / "_" / "~" / uchar
899  pct-encoded    = "%" HEXDIG HEXDIG
900  uchar         = %xA0-D7FF / %xF900-FDCF / %xFDF0-FFEF
901                / %x10000-1FFFFD / %x20000-2FFFFD / %x30000-3FFFFD
902                / %x40000-4FFFFD / %x50000-5FFFFD / %x60000-6FFFFD
903                / %x70000-7FFFFD / %x80000-8FFFFD / %x90000-9FFFFD
904                / %xA0000-AFFFFD / %xB0000-BFFFFD / %xC0000-CFFFFD
905                / %xD0000-DFFFFD / %xE1000-EFFFFD
906  reserved      = gen-delims / sub-delims
907  gen-delims     = ":" / "/" / "?" / "#" / "[" / "]" / "@"
908  sub-delims    = "! " / "$" / "&" / "'" / "(" / ")"
909                / "*" / "+" / "," / ";" / "="
910  unreserved    = ALPHA / DIGIT / "-" / "." / "_" / "~"

```

911 **Appendix B. Transforming HTTP IRIs to XRIs**
912 **(Non-Normative)**

913 To leverage existing infrastructure, it may sometimes be useful to convert HTTP IRIs into XRIs.
914 Because XRI syntax is, for the most part, a superset of generic IRI syntax, the majority of HTTP
915 IRIs can be converted to valid XRIs simply by replacing the scheme name “http” with “xri”.
916 Generally the authority component of the resulting XRI will be properly interpreted as an IRI
917 authority. There are some cases, however, in which a legal authority component in an IRI will be
918 interpreted as an XRI authority rather than an IRI authority when the IRI is converted to an XRI.
919 For example,

920 `http://!!1/example`

921 is a legal IRI. Converted to an XRI, it would become

922 `xri://!!1/example`

923 The authority “!!1” matches both the “xri-authority” and the “iauthority” ABNF productions. It would
924 be interpreted as an XRI authority, however, based on the “first-match-wins” rule used to resolve
925 ambiguities in the ABNF. Section 2.2.1.2 provides other examples of legal IRI authorities that
926 would be interpreted as XRI authorities when used in an XRI. Note, however, that these cases
927 are unlikely to arise in practice, since they typically result in an invalid URI when converted from
928 an IRI.

929 Special consideration must also be given to HTTP IRIs employing those characters in common
930 between the “sub-delims” production of **[IRI]** and the “xri-gen-delims” production of this
931 specification, namely opening parenthesis (“(“), closing parenthesis (“)”), star (“*”), bang (“!”),
932 dollar sign (“\$”), plus sign (“+”) and equals sign (“=”). These characters are reserved as delimiters
933 in HTTP IRIs but have no scheme-specific meaning (i.e., they are only used as delimiters in a
934 manner defined by a local authority). In XRIs, however, these characters do have defined
935 semantics that may or may not match the meaning intended by an IRI author. Conversion of such
936 IRIs to XRIs must be handled on a case-by-case basis.

937

Appendix C. Glossary

938 The following definitions are common to this specification, the *XRI Resolution* specification
939 **[XRIResolution]**, and the *XRI Metadata* specification **[XRIMetadata]**.) Note that this glossary
940 supercedes the glossary in **[XRIReqs]**.

941 **Absolute Identifier**

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

944 **Abstract Identifier**

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

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

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

950 A URN as described in **[RFC2141]** is one kind of abstract identifier. Compared to
951 concrete identifiers, abstract identifiers permit additional levels of indirection in
952 referencing resources, which can be useful for a variety of purposes, including
953 persistence, equivalence, human-friendliness, and data protection.

954 **Authority (or Identifier Authority)**

955 In the context of identifiers, an authority is a resource that assigns identifiers to other
956 resources. Note that in URI syntax as defined in **[URI]**, the “authority” production refers
957 explicitly to the top-level authority identified by the segment beginning with “/”. Since XRI
958 syntax supports unlimited federation, the term “authority” can technically refer to an
959 identifier authority at any level. However, in the “xri-authority” and “iauthority” productions
960 (section 2.2.1), it explicitly refers to the top-level identifier authority. See also “IRI
961 Authority” and “XRI Authority”

962

963 In the context of identifier resolution, an authority is a resource (typically a server) that
964 responds to resolution requests from another resource (typically a client). From this
965 perspective, each sub-segment in the authority segment of an XRI identifies a separate
966 authority.

967 **Base Identifier**

968 An absolute identifier that identifies a context for a relative identifier. Changing the base
969 identifier changes the context of the relative identifier. See “Relative Identifier.”

970 **Canonical Form**

971 The form of an identifier after applying transformation rules for the purpose of determining
972 equivalence. See also “Normal Form”.

973 **Community (or Identifier Community)**

974 A set of resources that share a common identifier authority, often (but not always) a
975 common root authority. Technically, a set of resources whose identifiers form a directed
976 graph or tree.

977 **Concrete Identifier**

978 An identifier that can be directly resolved to a resource or resource representation, rather
979 than to another identifier. Examples include the MAC address of a networked computer
980 and a phone number that rings directly to a specific device. All concrete identifiers are
981 intended to be resolvable. Contrast with “Abstract Identifier.”

982 **Context (or Identifier Context)**

983 The resource of which an identifier is an attribute. For example, in the string of identifiers
984 "a/b/c", the context of the identifier "b" is the resource identified by "a/", and the context of
985 the identifier "c" is the resource identified by "a/b/". Since multiple resources may assign
986 an identifier for a target resource, the resource can be said to be identified in multiple
987 contexts. For absolute identifiers, the context is global, i.e., there is a known starting
988 point, or root. For relative identifiers, the context is implicit. See also "Base Identifier."

989 **Cross-reference**

990 An identifier assigned in one context that is reused in another context. Cross-references
991 enable the expression of polyarchical relationships (relationships that cross multiple
992 hierarchies – see "Polyarchy".) Cross-references can be used to identify logically
993 equivalent resources in different domains, authorities, or physical locations. For example,
994 a cross-reference may be used to identify the same logical invoice stored in two
995 accounting systems (the originating system and the receiving system), the same logical
996 Web document stored on multiple proxy servers, the same logical datatype used in
997 multiple databases or XML schemas, or the same logical concept used in multiple
998 taxonomies or ontologies.

999 In XRI syntax, cross-references are syntactically delimited by enclosing them in
1000 parentheses. This is analogous to enclosing a word or phrase in quotation marks in a
1001 natural language, such as English, to indicate that the author is referring to it independent
1002 of the current context. For example, the phrase "love bird" is quoted in this sentence to
1003 indicate that we are *mentioning*, rather than *using*, the phrase - that is, we are referring to
1004 it independent of the context of this glossary.

1005 **Delegated Identifier**

1006 A multi-segment identifier in which segments are assigned by more than one identifier
1007 authority. Namespace authority is delegated from one identifier authority to the next.
1008 Mutually exclusive with "Local Identifier."

1009 **Federated Identifier**

1010 A delegated identifier that spans multiple independent identifier authorities. See also
1011 "Delegated Identifier."

1012 **Global Context Symbol (GCS)**

1013 A reserved character used at the start of the authority segment of an XRI to establish the
1014 global context of an XRI authority. XRI 2.0 defines four Global Context Symbols, which
1015 are used to represent persons, organizations, the public, and standards specifications.
1016 See section 2.2.1.2.

1017 **Hierarchy**

1018 A branching tree structure in which all primary relationships are parent-child. (Sibling
1019 relationships in a hierarchy are secondary, derived from the parent-child relationships.)
1020 URI and IRI syntax has explicit support for hierarchical paths. XRI syntax supports both
1021 hierarchical and polyarchical paths. See "Polyarchy" and "Cross-reference."

1022 **Human-Friendly Identifier (HFI)**

1023 An identifier containing words or phrases intended to convey meaning in a specific
1024 human language which is therefore easy for people to remember and use. Contrast with
1025 "Machine-Friendly Identifier."

1026 **Identifier**

1027 Per [URI], anything that "embodies the information required to distinguish what is being
1028 identified from all other things within its scope of identification." In UML terms, an
1029 identifier is an attribute of a resource (the identifier context) that forms an association with

1030 another resource (the identifier target). The general term “identifier” does not specify
1031 whether the identifier is abstract or concrete, absolute or relative, persistent or
1032 reassignable, human-friendly or machine-friendly, delegated or local, hierarchical or
1033 polyarchical, or resolvable or self-referential.

1034 **I-name**

1035 An informal term used to refer to a reassignable XRI; more specifically, an XRI in which
1036 at least one sub-segment is reassignable.

1037 **I-number**

1038 An informal term used to refer to a persistent XRI; more specifically, an XRI in which all
1039 sub-segments are persistent. Note that a persistent XRI is not required to be numeric - it
1040 may be any text string meeting the XRI ABNF requirements.

1041 **IRI (Internationalized Resource Identifier)**

1042 IRI is a specification for internationalized URIs developed by the W3C. IRIs specify how
1043 to include characters from the Universal Character Set (Unicode/ISO10646) in URIs. The
1044 IRI specification **[IRI]** provides a mapping from IRIs to URIs, which allows IRIs to be used
1045 instead of URIs where appropriate. This XRI specification defines a similar transformation
1046 from XRIs to IRIs for the same reason.

1047 **IRI Authority**

1048 An identifier authority (see “Authority”) represented by the authority segment of an XRI
1049 that does not match the “xri-authority” production but matches the “iauthority” production
1050 from **[IRI]**. See section 2.2.1.3. Mutually exclusive with “XRI Authority”.

1051 **Local Identifier**

1052 Any identifier, or any set of segments in a multi-segment identifier, that is assigned by the
1053 same identifier authority. Each of these segments is local to that authority. Mutually
1054 exclusive with “Delegated Identifier.”

1055 **Machine-Friendly Identifier (MFI)**

1056 An identifier containing digits, hexadecimal values, or other character sequences
1057 optimized for efficient machine indexing, searching, routing, caching, and resolvability.
1058 MFIs generally do not contain human semantics. Compare with “Human-Friendly
1059 Identifier.”

1060 **Normal Form**

1061 The character-by-character format of an identifier after encoding, escaping, or other
1062 character transformation rules have been applied in order to satisfy syntactic
1063 requirements. Three normal forms are defined for XRIs—XRI-normal form, IRI-normal
1064 form, and URI-normal form. See section 2.3.1 for details. See also “Canonical Form”.

1065 **Path**

1066 The relationships between resources defined by a multi-segment identifier. In less strict
1067 contexts, the word “path” often refers to the multi-segment identifier itself, or to the
1068 resources it represents (such as filesystem directories).

1069 **Persistent Identifier**

1070 An identifier that is permanently assigned to a resource and intended never to be
1071 reassigned to another resource - even if the original resource goes off the network, is
1072 terminated, or ceases to exist. A URN as described in **[RFC2141]** is an example of a
1073 persistent identifier. Persistent identifiers tend to be machine-friendly identifiers, since
1074 human-friendly identifiers often reflect human semantic relationships that may change
1075 over time. Mutually exclusive with “Reassignable Identifier.”

- 1076 **Polyarchy**
1077 A treelike structure composed of multiple intersecting hierarchies in which primary
1078 relationships can cross hierarchies. A polyarchy allows one member to be connected or
1079 linked to any other, although, in contrast to a web, the overall structure tends to remain
1080 strongly hierarchical. XRIs support polyarchic paths through the use of cross-references.
1081 See also “Cross-reference” and “Hierarchy”.
- 1082 **Reassignable Identifier**
1083 An identifier that may be reassigned from one resource to another. Example: the domain
1084 name “example.com” may be reassigned from ABC Company to XYZ Company, or the
1085 email address “mary@example.com” may be reassigned from Mary Smith to Mary Jones.
1086 Reassignable identifiers tend to be human-friendly because they often represent the
1087 potentially transitory mapping of human semantic relationships onto network resources or
1088 resource representations. Mutually exclusive with “Persistent Identifier.”
- 1089 **Relative Identifier**
1090 An identifier that refers to a resource only in relationship to a particular context (for
1091 example, the current community, the current document, or the current position in a
1092 delegated identifier). If the context changes, the identifier’s meaning also changes. A
1093 relative identifier can be converted into an absolute identifier by combining it with a base
1094 identifier (an absolute identifier that is used to identify a context). See “Base Identifier”.
1095 Mutually exclusive with “Absolute Identifier.”
- 1096 **Resolvable Identifier**
1097 An identifier that references a network resource or resource representation and that can
1098 be resolved into a network endpoint for communicating with the target resource. Mutually
1099 exclusive with “Self-Reference.”
- 1100 **Resource**
1101 Per [URI], “anything that can be named or described.” Resources are of two types:
1102 network resources (those that are network-addressable) and non-network resources
1103 (those that exist entirely independent of a network). Network resources are themselves of
1104 two types: physical resources (resources physically attached to or operating on the
1105 network) or resource representations (see “Resource Representation”).
- 1106 **Resource Representation**
1107 A network resource that represents the attributes of another resource. A resource
1108 representation may represent either another network resource (such as a machine,
1109 service, or application) or a non-network resource (such as a person, organization, or
1110 concept).
- 1111 **Segment (or Identifier Segment)**
1112 Any syntactically delimited component of an identifier. In generic URI syntax, all
1113 segments after the authority portion are delimited by forward slashes
1114 (“/segment1/segment2/...”). In XRI syntax, slash segments can be further subdivided into
1115 sub-segments called *star segments* (for reassignable identifiers) and *bang segments* (for
1116 persistent identifiers). See section 2.2.3. XRI also supports another type of segment
1117 called a cross-reference, which is enclosed in parentheses. See “Cross-Reference”.
- 1118 **Self-Reference (or Self-Referential Identifier)**
1119 An identifier which is itself the representation of the resource it references. Self-
1120 references are typically used to represent non-network resources (e.g., “love”, “Paris”,
1121 “the planet Jupiter”) in contexts where this identifier is not intended to be resolved to a
1122 separate network representation of that resource. The primary purpose of self-references
1123 is to establish equivalence across contexts (see “Cross-References”). Mutually exclusive
1124 with “Resolvable Identifier.”

- 1125 **Sub-segment**
- 1126 A syntactically delimited component of an identifier segment (see “Segment”). While URI
1127 and IRI syntax define only segments, XRI syntax defines both segments and sub-
1128 segments. XRI sub-segments are used to distinguish among persistent identifiers,
1129 reassignable identifiers, and cross-references. See sections 2.2.2 and 2.2.3.
- 1130 **Synonym (or Identifier Synonym)**
- 1131 An identifier that is asserted by an identifier authority to be equivalent to another identifier
1132 not because of strict literal equivalence, but because it resolves to the same resource.
- 1133 **Target (or Identifier Target)**
- 1134 The resource referenced by an identifier. A target may be either a network resource
1135 (including a resource representation) or a non-network resource.
- 1136 **URI (Uniform Resource Identifier)**
- 1137 The standard identifier used in World Wide Web architecture. Starting in 1998, RFC 2396
1138 has been the authoritative specification for URI syntax. In January 2005 it was
1139 superseded by RFC 3986 **[URI]**.
- 1140 **XDI (XRI Data Interchange)**
- 1141 A generalized, extensible service for sharing, linking, and synchronizing XML data and
1142 metadata associated with XRI-identified resources. XDI is being developed by the OASIS
1143 XDI Technical Committee (<http://www.oasis-open.org/committees/xdi>).
- 1144 **XRI Authority**
- 1145 An identifier authority (see “Authority”) represented by the authority segment of an XRI
1146 that begins with either a global context symbol or a cross-reference. See section 2.2.1.1.
1147 Mutually exclusive with “IRI Authority.”
- 1148 **XRI Descriptor (XRID)**
- 1149 An XML document returned by an authority in the process of XRI resolution as defined in
1150 section 2.2.2 of the *XRI Resolution* specification **[XRIResolution]**.
- 1151 **XRI Reference**
- 1152 A term that includes both absolute and relative XRIs. Used in the same way as “URI
1153 reference” and “IRI reference.” Note that to transform an XRI reference into an XRI, it
1154 must first be converted to absolute form, which in the case of a relative XRI requires the
1155 use of a base XRI to establish context.

1156

Appendix D. Acknowledgments

1157 The editors would like to acknowledge the contributions of the OASIS XRI Technical Committee,
1158 whose voting members at the time of publication were:

- 1159 • Geoffrey Strongin, Advanced Micro Devices
- 1160 • Ajay Madhok, AmSoft Systems
- 1161 • Jean-Jacques Dubray, Attachmate
- 1162 • William Barnhill, Booz Allen and Hamilton
- 1163 • Drummond Reed, Cordance Corporation
- 1164 • Marc Le Maitre, Cordance Corporation
- 1165 • Dave McAlpin, Epok
- 1166 • Loren West, Epok
- 1167 • Peter Davis, NeuStar
- 1168 • Masaki Nishitani, Nomura Research
- 1169 • Nat Sakimura, Nomura Research
- 1170 • Tetsu Watanabe, Nomura Research
- 1171 • Owen Davis, PlaNetwork
- 1172 • Victor Grey, PlaNetwork
- 1173 • Fen Labalme, PlaNetwork
- 1174 • Mike Lindelsee, Visa International
- 1175 • Gabriel Wachob, Visa International
- 1176 • Dave Wentker, Visa International
- 1177 • Bill Washburn, XDI.ORG

1178 The editors also would like to acknowledge the following people for their contributions to previous
1179 versions of the OASIS XRI specifications (affiliations listed for OASIS members):

1180 Thomas Bikeev, EAN International; Krishna Sankar, Cisco; Winston Bumpus, Dell; Joseph
1181 Moeller, EDS; Steve Green, Epok; Lance Hood, Epok; Adarbad Master, Epok; Davis McPherson,
1182 Epok; Chetan Sabnis, Epok; Phillipe LeBlanc, GemPlus; Jim Schreckengast, Gemplus; Xavier
1183 Serret, Gemplus; John McGarvey, IBM; Reva Modi, Infosys; Krishnan Rajagopalan, Novell;
1184 Tomonori Seki, NRI; James Bryce Clark, OASIS; Marc Stephenson, TSO; Mike Mealling,
1185 Verisign; Rajeev Maria, Visa International; Terence Spielman, Visa International; John Veizades,
1186 Visa International; Lark Allen, Wave Systems; Michael Willett, Wave Systems; Matthew Dovey;
1187 Eamonn Neylon; Mary Nishikawa; Lars Marius Garshol; Norman Paskin; Bernard Vatant.

1188 A special acknowledgement to Jerry Kindall (Epok) for a full editorial review.

1189 Also, the authors of and contributors to the following documents and specifications are
1190 acknowledged for the intellectual foundations of the XRI specification:

- 1191 • RFC 1737
- 1192 • RFC 2616
- 1193 • RFC 2718
- 1194 • RFC 3986 (STD 66) and its predecessor, RFC 2396
- 1195 • RFC 3987
- 1196 • XNS
- 1197

1198

Appendix E. Notices

1199 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
1200 that might be claimed to pertain to the implementation or use of the technology described in this
1201 document or the extent to which any license under such rights might or might not be available;
1202 neither does it represent that it has made any effort to identify any such rights.

1203 Information on OASIS's procedures with respect to rights in OASIS specifications can be found at
1204 the OASIS website. Copies of claims of rights made available for publication and any assurances
1205 of licenses to be made available, or the result of an attempt made to obtain a general license or
1206 permission for the use of such proprietary rights by implementors or users of this specification,
1207 can be obtained from the OASIS President.

1208 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
1209 applications, or other proprietary rights which may cover technology that may be required to
1210 implement this specification. Please address the information to the OASIS President.

1211 **Copyright © OASIS Open 2005. All Rights Reserved.**

1212 This document and translations of it may be copied and furnished to others, and derivative works
1213 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
1214 published and distributed, in whole or in part, without restriction of any kind, provided that the
1215 above copyright notice and this paragraph are included on all such copies and derivative works.
1216 However, this document itself does not be modified in any way, such as by removing the
1217 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
1218 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
1219 Property Rights document must be followed, or as required to translate it into languages other
1220 than English.

1221 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
1222 successors or assigns.

1223 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1224 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO
1225 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE
1226 ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1227 PARTICULAR PURPOSE.