1

Extensible Resource Identifier (XRI) Syntax V2.0

2

3

Committee Draft 02, 13 October 2005

4

9 **Editors:**
10   Drummond Reed, Cordance <drummond.reed@cordance.net>
11   Dave McAlpin, Epok <dave.mcalpin@epok.net>

12 **Contributors:**
13   Peter Davis, Neustar <peter.davis@neustar.biz>
14   Nat Sakimura, NRI <n-sakimura@nri.co.jp>
15   Mike Lindelsee, Visa International <mlindels@visa.com>
16   Gabe Wachob, Visa International <gwachob@visa.com>

17 **Abstract:**
18   This document is the normative technical specification for XRI generic syntax. For a non-
19   normative introduction to the uses and features of XRIs, see *Introduction to XRIs*
20   **[XRIIntro]**.

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

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

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

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

Table of Contents

Introduction

1.1 Overview of XRIs

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

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

Abstract	XRI	Extends syntax
Internationalized	IRI	Extends character set
Uniform	URI	Base specification

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

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

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

This document specifies the normative syntax for XRIs, along with associated normalization, processing and equivalence rules. See also *An Introduction to XRIs* **[XRIIntro]** for a non-normative introduction to XRI architecture.

1.1.1 Generic Syntax

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

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

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

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

1. *Persistent and reassignable segments.* Unlike generic URI syntax, XRI syntax allows the internal components of an XRI reference to be explicitly designated as either persistent or reassignable.

128 2. *Cross-references.* Cross-references allow XRI references to contain other XRI references
129 or IRIs as syntactically-delimited sub-segments. This provides syntactic support for
130 "compound identifiers", i.e., the use of well-known, fully-qualified identifiers within the
131 context of another XRI reference. Typical uses of cross-references include using well-
132 known types of metadata in an XRI reference (such as language or versioning metadata),
133 or the use of globally-defined identifiers to mark parts of an XRI reference as having
134 application- or vocabulary-specific semantics.

135 3. *Additional authority types.* While XRI syntax supports the same generic syntax used in
136 IRIs for DNS and IP authorities, it also provides two additional options for identifying an
137 authority: a) global context symbols (GCS), shorthand characters used for establishing
138 the abstract global context of an identifier, and b) cross-references, which enable any
139 identifier to be used to specify an XRI authority.

140 4. *Standardized federation.* Federated identifiers are those delegated across multiple
141 authorities, such as DNS names. Generic URI syntax leaves the syntax for federated
142 identifiers up to individual URI schemes, with the exception of explicit support for IP
143 addresses. XRI syntax standardizes federation of both persistent and reassignable
144 identifiers at any level of the path.

1.1.2 URI, URL, URN, and XRI

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

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

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

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

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

160 "An individual [URI] scheme does not have to be classified as being just one of 'name' or
161 'locator'. Instances of URIs from any given scheme may have the characteristics of names or
162 locators or both, often depending on the persistence and care in the assignment of identifiers
163 by the naming authority, rather than on any quality of the scheme."

164 XRIs are consistent with this philosophy. Although XRIs are designed to fulfill the requirements of
165 abstract "names" that are resolved into concrete locators, XRI syntax does not distinguish
166 between identifiers that represent "names", "locators" or "characteristics."

1.2 Terminology and Notation

1.2.1 Keywords

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

1.2.2 Syntax Notation

173

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

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

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

187
```
    This is an example of ABNF specific to XRI.
```

188
```
    This is an example of ABNF inherited from IRI.
```

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

2 Syntax

191

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

2.1 Characters

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

2.1.1 Character Encoding

200 The standard character encoding of XRI is UTF-8, as recommended by **[RFC2718]**. When an XRI
201 reference is presented as a human-readable identifier, the representation of the XRI reference in
202 the underlying document may use the character encoding of the underlying document. However,
203 this representation must be converted to UTF-8 before the XRI can be processed outside the
204 document. This encoding in UTF-8 MUST include normalization according to Normalization Form
205 KC (NFKC) as defined in **[UTR15]**. The stricter NFKC is specified rather than Normalization Form
206 C (NFC) used in IRI encoding **[IRI]** because NFKC reduces the number of UCS compatability
207 characters allowed in an XRI and increases the probability of equivalence matches.

2.1.2 Reserved Characters

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

```
216     xri-reserved    = xri-gen-delims / xri-sub-delims

217     xri-gen-delims  = ":" / "/" / "?" / "#" / "[" / "]" / "(" / ")"
218                     / "*" / "!" / rgcs-char

219     xri-sub-delims  = "&" / ";" / "," / "'"
```

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

2.1.3 Unreserved Characters

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

```
226     iunreserved        = ALPHA / DIGIT / "-" / "." / "_" / "~" / ucschar
```

```
227    ucschar            = %xA0-D7FF / %xF900-FDCF / %xFDF0-FFEF
228                       / %x10000-1FFFD / %x20000-2FFFD / %x30000-3FFFD
229                       / %x40000-4FFFD / %x50000-5FFFD / %x60000-6FFFD
230                       / %x70000-7FFFD / %x80000-8FFFD / %x90000-9FFFD
231                       / %xA0000-AFFFD / %xB0000-BFFFD / %xC0000-CFFFD
232                       / %xD0000-DFFFD / %xE1000-EFFFD
```

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

2.1.4 Percent-Encoded Characters

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

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

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

```
250    pct-encoded    = "%" HEXDIG HEXDIG
```

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

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

2.1.4.1 Encoding XRI Metadata

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

263 XRI syntax offers an option for encoding this language metadata using a cross-reference
264 beginning with the GCS "$" symbol (see section 2.2.1.2). The top level authority for language
265 metadata is the *XRI Metadata Specification* published by the OASIS XRI Technical Committee.

2.1.5 Excluded Characters

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

272 Note that presenting "space" or other whitespace characters in a non-percent-encoded form is not
273 recommended for several reasons. First, it is often difficult to visually determine the number of
274 spaces or other characters composing a block of whitespace, leading to transcription errors.
275 Second, the space character is often used to delimit an XRI reference, so non-percent-encoded
276 whitespace characters can make it difficult or impossible to determine where the identifier ends.
277 Finally, non-percent-encoded whitespace can be used to maliciously construct subtly different
278 identifiers intended to mislead the reader. For these reasons, non-percent-encoded whitespace
279 characters SHOULD be avoided in presentation, and alternatives to whitespace as a logical
280 separator within XRIs (such as dots or hyphens) SHOULD be used whenever possible.

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

283 "The UCS contains many areas of characters for which there are strong visual
284 look-alikes. Because of the likelihood of transcription errors, these also should be
285 avoided. This includes the full-width equivalents of Latin characters, half-width
286 Katakana characters for Japanese, and many others. This also includes many
287 look-alikes of 'space', 'delims', and 'unwise', characters excluded in **[RFC3491]**."

288 "Additional information is available from **[UniXML]**. **[UniXML]** is written in the
289 context of running text rather than in the context of identifiers. Nevertheless, it
290 discusses many of the categories of characters not appropriate for IRIs."

291 Finally, although they are not excluded characters, special care should be taken by user agents
292 with regard to the display of UCS characters that are visual look-alikes (homographs) for XRI
293 delimiters (all characters in the xri-reserved production, section 2.1.2). See section 3.3, "Spoofing
294 and Homographic Attacks" for additional information.

2.2 Syntax Components

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

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

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

307    xri-hier-part    = ( xri-authority / iauthority ) xri-path-abempty

308    XRI-reference    = XRI / relative-XRI-ref

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

310    relative-XRI-ref = relative-XRI-part [ "?" iquery ] [ "#" ifragment ]

311    relative-XRI-part = xri-path-absolute
312                       / xri-path-noscheme
313                       / ipath-empty

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

315    xri-no-scheme    = xri-hier-part [ "?" iquery ] [ "#" ifragment ]
```

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

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

323 2.2.1 Authority

324 XRIs support the same types of authorities as generic IRIs, called *IRI authorities.* XRIs also
325 support additional types of abstract identification authorities called *XRI authorities.*

326 2.2.1.1 XRI Authority

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

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

330 2.2.1.2 Global Context Symbol (GCS) Authority

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

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

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

335    rgcs-authority    = rgcs-char xri-segment

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

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

342

Symbol Character	Authority Type	Establishes Global Context For
=	Person	Identifiers for whom the authority is controlled by an individual person.
@	Organi-zation	Identifiers for whom the authority is controlled by an organization or a resource in an organizational context.
+	General public	Identifiers for whom there is no specific controlling authority because they represent generic dictionary concepts or "tags" whose meaning is determined by consensus. (In the English language, for example, these would be the generic nouns.)
$	Standards body	Identifiers for whom the authority is controlled by a specification from a standards body, for example, other XRI specifications from the OASIS XRI Technical Committee, other OASIS specifications, or (using cross-references) other standards bodies.

343   Table 1: XRI global context symbols.

2.2.1.3 IRI Authority

345   XRIs support the same type of authority defined by the "iauthority" production of **[IRI]**.

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

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

348     ihost             = IP-literal / IPv4address / ireg-name

349     port              = *DIGIT
```

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

354   A host identifier can be followed by an optional port number. The XRI syntax specification does
355   not define a default port because it is expected this will be inherited from the resolution protocol.
356   Therefore, if the port is omitted in an XRI, it is undefined.

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

2.2.2 Cross-References

363   Cross-references are the primary extensibility mechanism in XRI. They allow an identifier
364   assigned in one context to be reused in another context, permitting identifiers to be shared across
365   contexts. This simplifies identifying logically equivalent resources across hierarchies (a directory
366   concept referred to as "polyarchy".)

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

370
```
xref              = "(" ( XRI-reference / IRI ) ")"
```

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

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

376
```
xref-authority    = xref *xri-subseg
```

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

379
380
```
xri://(mailto:john.doe@example.com)/favorites/home
      --example of using a URI as an XRI global authority
```

381 2.2.3 Path

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

388
389
390
391
```
xri-path          = xri-path-abempty
                  / xri-path-absolute
                  / xri-path-noscheme
                  / ipath-empty
```

392
```
xri-path-abempty  = *( "/" xri-segment )
```

393
```
xri-path-absolute = "/" [ xri-segment-nz *( "/" xri-segment ) ]
```

394
395
```
xri-path-noscheme = xri-subseg-od-nx *xri-subseg-nc
                    *( "/" xri-segment )
```

396
```
xri-segment       = xri-subseg-od *xri-subseg
```

397
```
xri-segment-nz    = xri-subseg-od-nz *xri-subseg
```

398
```
xri-subseg        = ( "*" / "!" ) (xref / *xri-pchar)
```

399
```
xri-subseg-nc     = ( "*" / "!" ) (xref / *xri-pchar-nc)
```

400
```
xri-subseg-od     = [ "*" / "!" ] (xref / *xri-pchar)
```

401
```
xri-subseg-od-nz  = [ "*" / "!" ] (xref / 1*xri-pchar)
```

402
```
xri-subseg-od-nx  = [ "*" / "!" ] 1*xri-pchar-nc
```

403
```
xri-subseg-pt-nz  = "!" (xref / 1*xri-pchar)
```

404 * segments are used to specify *reassignable identifiers*—identifiers that may be reassigned by an
405 identifier authority to represent a different resource at some future date. ! segments are used to
406 specify *persistent identifiers*—identifiers that are permanently assigned to a resource and will not
407 be reassigned at a future date. A ! segment SHOULD meet the requirements for persistent
408 identifiers set out in *Functional Requirements for Uniform Resource Names* **[RFC1737]**. The
409 default is a * segment, so a leading star ("*") is optional for the first (or only) sub-segment if this
410 subsegment is reassignable.

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
```
    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.

2.2.4 Query

424 The XRI query component is identical to the IRI query component as described in section 2.2 of
425 **[IRI]**.

426
```
    iquery            = *( ipchar / iprivate / "/" / "?" )
```

2.2.5 Fragment

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

429
```
    ifragment         = *( ipchar / "/" / "?" )
```

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

2.3 Transformations

2.3.1 Transforming XRI References into IRI and URI References

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

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

443 Applications transforming XRI references to IRI references MUST use the following steps (or a
444 process that achieves exactly the same result). Before applying these steps, the XRI reference
445 must be in XRI-normal form as defined in section 2.1.4.

446　1. If the XRI reference is not encoded in UTF-8, convert the XRI reference to a sequence of
447　　　characters encoded in UTF-8, normalized according to Normalization Form KC (NFKC)
448　　　as defined in **[UTR15]**.

449　2. If the XRI reference is not relative (i.e., if it matches the "XRI" ABNF production) and the
450　　　optional "xri://" prefix has been omitted, prepend "xri://" to the XRI reference.

451　3. Optionally add XRI metadata using cross-references as defined in section 2.1.4.1. Note
452　　　that the addition of XRI metadata may change the resulting IRI or URI reference for the
453　　　purposes of comparison as explained in section 2.5.4.

454　4. Apply the XRI escaping rules defined in section 2.3.2. Note that this step is not
455　　　idempotent (i.e., it may yield a different result if applied more than once), so it is very
456　　　important that implementers not apply this step more than once to avoid changing the
457　　　semantics of the identifier.

458　At the completion of step 4, the percent-encoded XRI reference is now in IRI-normal form and
459　may be used as an IRI reference conformant with **[IRI]**.

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

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

2.3.2 Escaping Rules for XRI Syntax

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

471　The first rule deals with cross-references as explained in section 2.2.2. Since a cross-reference
472　contains either an IRI or an XRI reference (which itself may contain further nested IRIs or XRI
473　references), it may include characters that, if not escaped, would cause misinterpretation when
474　the XRI reference is used in a context that expects an IRI or URI reference. Consider the
475　following XRI:

476
```
xri://@example/(xri://@example2/abc?id=1)
```

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

479
```
scheme = xri
authority = @example
path = /(xri://@example2/abc
query = id=1)
```
480
481
482

483　The desired separation is:

484
```
scheme = xri
authority = @example
path = /(xri://@example2/abc?id=1)
query = <undefined>
```
485
486
487

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

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

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

494 In addition, the slash "/" character in a cross-reference may also be misinterpreted by a non-XRI-
495 aware parser. Consider:

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

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

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

500 instead of the intended:

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

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

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

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

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

510 This slash character between "c" and "d" is percent-encoded to show that it's not a syntactical
511 element of the XRI, i.e., that it should be interpreted as data and not as a delimiter. To preserve
512 this type of distinction when converting an XRI reference to an IRI reference, the percent "%"
513 character must be percent-encoded as "%25". Following this rule, the above example fully
514 converted would be:

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

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

519 1. Percent-encode all percent "%" characters as "%25" across the entire XRI reference.
520 2. Percent-encode all number sign "#" characters that appear within a cross-reference as
521 "%23".
522 3. Percent-encode all question mark "?" characters that appear within a cross-reference as
523 "%3F".
524 4. Percent-encode all slash "/" characters that appear within a cross-reference as "%2F".

2.3.3 Transforming IRI References into XRI References

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

528 1. If the XRI reference is not encoded in UTF-8, convert the XRI reference to a sequence of
529 characters encoded in UTF-8, normalized according to Normalization Form KC (NFKC)
530 as defined in **[UTR15]**.

531   2. Perform the following special conversions for XRI syntax:

532   a. Convert all percent-encoded slash ("/") characters to their corresponding octets.

533   b. Convert all percent-encoded question mark ("?") characters to their
534   corresponding octets.

535   c. Convert all percent-encoded number sign ("#") characters to their corresponding
536   octets.

537   d. Convert all percent-encoded percent ("%") characters to their corresponding
538   octets.

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

2.4 Relative XRI References

2.4.1 Reference Resolution

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

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

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

559 • treats the characters allowed in IRI references but not in URI references the same as it
560   treats unreserved characters in URI references (as required by section 5 of **[IRI]**) and

561 • treats all characters within all cross-references the same as unreserved characters in URI
562   references (i.e., treats cross-references as opaque with respect to relative reference
563   resolution).

2.4.2 Reference Resolution Examples

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

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

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

2.4.2.1 Normal Examples

```
571     !g!g        =   xri://@a*a/!b!b/c*c/!g!g
572     ./!g!g      =   xri://@a*a/!b!b/c*c/!g!g
573     !g!g/       =   xri://@a*a/!b!b/c*c/!g!g/
574     /!g!g       =   xri://@a*a/!g!g
575     //@!g!g     =   Not a legal relative XRI reference
576     ?y          =   xri://@a*a/!b!b/c*c/(xri://@d*d/e)?y
577     !g!g?y      =   xri://@a*a/!b!b/c*c/!g!g?y
```

```
578        #s          =   xri://@a*a/!b!b/c*c/(xri://@d*d/e)?q#s
579        !g!g#s      =   xri://@a*a/!b!b/c*c/!g!g#s
580        !g!g?y#s    =   xri://@a*a/!b!b/c*c/!g!g?y#s
581        ;x          =   xri://@a*a/!b!b/c*c/;x
582        !g!g;x      =   xri://@a*a/!b!b/c*c/!g!g;x
583        !g!g;x?y#s  =   xri://@a*a/!b!b/c*c/!g!g;x?y#s
584                    =   xri://@a*a/!b!b/c*c/(xri://@d*d/e)?q
585        .           =   xri://@a*a/!b!b/c*c/
586        ./          =   xri://@a*a/!b!b/c*c/
587        ..          =   xri://@a*a/!b!b/
588        ../         =   xri://@a*a/!b!b/
589        ../!g!g     =   xri://@a*a/!b!b/!g!g
590        ../..       =   xri://@a*a/
591        ../../      =   xri://@a*a/
592        ../../!g!g  =   xri://@a*a/!g!g
```

2.4.2.2 Abnormal Examples

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

```
595        ../../../!g!g    =   xri://@a*a/!g!g
596        ../../../../!g!g =   xri://@a*a/!g!g
```

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

```
599        /./!g!g     =   xri://@a*a/!g!g
600        /../!g!g    =   xri://@a*a/!g!g
601        !g!g.       =   xri://@a*a/!b!b/c*c/!g!g.
602        .!g!g       =   xri://@a*a/!b!b/c*c/.!g!g
603        !g!g..      =   xri://@a*a/!b!b/c*c/!g!g..
604        ..!g!g      =   xri://@a*a/!b!b/c*c/..!g!g
```

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

```
607        ./../!g!g       =   xri://@a*a/!b!b/!g!g
608        ./!g!g/.        =   xri://@a*a/!b!b/c*c/!g!g/
609        !g!g/./h        =   xri://@a*a/!b!b/c*c/!g!g/h
610        !g!g/../h        =   xri://@a*a/!b!b/c*c/h
611        !g!g;x=1/./y    =   xri://@a*a/!b!b/c*c/!g!g;x=1/y
612        !g!g;x=1/../y   =   xri://@a*a/!b!b/c*c/y
```

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

```
616        !g!g?y/./x      =   xri://@a*a/!b!b/c*c/!g!g?y/./x
617        !g!g?y/../x     =   xri://@a*a/!b!b/c*c/!g!g?y/../x
618        !g!g#s/./x      =   xri://@a*a/!b!b/c*c/!g!g#s/./x
619        !g!g#s/../x     =   xri://@a*a/!b!b/c*c/!g!g#s/../x
```

2.4.3 Leading Segments Containing a Colon

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

> "A path segment that contains a colon character (e.g., 'this:that') cannot be used
> as the first segment of a relative-path reference because it would be mistaken for

625   a scheme name. Such a segment must be preceded by a dot-segment (e.g.,
626   './this:that') to make a relative-path reference."

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

2.4.4 Leading Segments Beginning with a Cross-Reference

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

2.5 Normalization and Comparison

636 In general, the normalization and comparison rules for generic IRIs and URIs specified in Section
637 5 of **[IRI]** and Section 6 of **[URI]** apply to XRIs. This section describes a number of additional XRI-
638 specific rules for normalization and comparison. To reduce the requirements imposed upon a
639 minimally conforming processor, the majority of these rules are RECOMMENDED rather than
640 REQUIRED. An implementation that fails to observe them, however, may frequently treat two
641 XRIs as non-equal when in fact they are equal.

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

2.5.1 Case

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

648 •   Comparison of the scheme component of XRIs and all IRIs used as cross-references is case-
649   insensitive.

650 •   Comparison of authority components (section 2.2.1) is case-insensitive as defined in **[IRI]**.

651 •   As specified in section 2.1.4, comparison of characters in a percent-encoding construction is
652   case-insensitive for the hexadecimal digits "A" through "F", i.e. "%ab" is equivalent to "%AB".

2.5.2 Encoding, Percent-Encoding, and Transformations

654 •   Two XRIs MUST be considered equivalent if they are character-for-character equivalent.
655   Therefore, they are also equivalent if they are byte-for-byte equivalent and use the same
656   character encoding.

657 •   Two XRIs that differ only in whether unreserved characters are percent-encoded SHOULD be
658   considered equivalent. If one XRI percent-encodes one or more unreserved characters, and
659   another XRI differs only in that the same characters are not percent-encoded, they are
660   equivalent.

661 •   All forms of an XRI during the transformation process described in section 2.3.1 SHOULD be
662   considered equivalent, assuming the same XRI metadata is inserted as described in section
663   2.3.1.

2.5.3 Optional Syntax

665 •   An "xri-segment" (section 2.2.3) that omits the optional leading star ("*") SHOULD be
666   considered equivalent to the same "xri-segment" prefixed with an star. For example the
667   segment "/foo*bar" is equivalent to the segment "/*foo*bar".

2.5.4 Cross-References

668

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

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

674 • While cross-references beginning with the GCS "$" symbol MAY be considered significant in
675   all cases, the specification governing a particular $ namespace MAY declare that cross-
676   references in that namespace should be ignored for purposes of comparison. Failure to follow
677   such a rule may lead to false negatives. See section 2.1.4.1.

2.5.5 Canonicalization

678

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

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

688 • The optional "xri://" prefix is included,

689 • The scheme is specified in lowercase,

690 • The authority component is specified in lowercase,

691 • Percent-encoding uses uppercase A through F,

692 • If optional, the leading star in xri-segments is omitted,

693 • Unnecessary percent-encoding is not present,

694 • ./ and ../ are absent in absolute XRIs, and

695 • Cross-references are reasonably canonical with respect to their schemes.

696 Table 2 illustrates the application of these rules. Although the XRIs in the first and second
697 columns are equivalent, the form in the second column is recommended.

698

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

699 Table 2: Examples of XRI canonicalization recommendations.

3 Security and Data Protection Considerations

700

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

704 - Reliability and Consistency

705 - Malicious Construction

706 - Back-End Transcoding

707 - Rare IP Address Formats

708 - Sensitive Information

709 - Semantic Attacks

710 This material notes that "a URI does not in itself pose a direct security threat." The same is true
711 of an XRI. However infrastructure and applications that use XRIs may have special security and
712 data protection considerations as noted in this section.

3.1 Cross-References

713

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

3.2 XRI Metadata

716

717 The use of cross-references employing the GCS "$" symbol for encoding XRI metadata in an XRI
718 (section 2.1.4.1) may involve other security and data protection considerations that are outside
719 the scope of this specification. These considerations SHOULD be addressed in the relevant $
720 namespace specification.

3.3 Spoofing and Homographic Attacks

721

722 One particularly important security consideration is spoofing, covered first in **[URI]** and more
723 thoroughly in **[IRI]** Section 7.5. Spoofing is a semantic attack in which an identifier is deliberately
724 constructed to deceive the user into believing it represents one resource when in fact it
725 represents another. With IRIs in particular, a common example of such an attack is using
726 characters from different scripts that are visual lookalikes ("homographs"), e.g., the Latin "A", the
727 Greek "Alpha", and the Cyrillic "A". Another common attack is using homographs of the delimiter
728 character "/" to deceive the user about the true contents of an IRI authority segment.

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

734 To help prevent this problem, XRI registries SHOULD institute policies preventing the registration
735 of deceptive XRIs. In addition, XRIs that use an XRI authority (section 2.2.1.1) are subject to a
736 particular semantic attack: spoofing the leading GCS character (section 2.2.1.2) with a
737 homograph from the Unicode character set. Such a character may cause users to believe they
738 are dealing with an XRI authority when in fact their user agent interprets the authority segment as
739 an IRI authority (section 2.2.1.3).

740 To help prevent this or any other attack based on spoofing legitimate XRI delimiters (all
741 characters in the xri-reserved production, section 2.1.2), user agents SHOULD employ one or
742 more of the following safeguards, particularly with regard to the authority segment of an XRI: a)

743 visually distinguish the defined XRI delimiter characters using special color, size, font, or other
744 mechanism that enables users to clearly understand when a legitimate XRI delimiter character is
745 being displayed, b) do not display any homograph of any XRI delimiter character in unencoded
746 form, and/or c) warn the user when an XRI contains a potentially deceptive homographic
747 character.

3.4 UTF-8 Attacks

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

751 "Implementers of UTF-8 need to consider the security aspects of how they
752 handle illegal UTF-8 sequences. It is conceivable that in some circumstances an
753 attacker would be able to exploit an incautious UTF-8 parser by sending it an
754 octet sequence that is not permitted by the UTF-8 syntax."

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

3.5 XRI Usage in Evolving Infrastructure

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

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

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

4 References

4.1 Normative

[IRI]	M. Dürst, M. Suignard, *Internationalized Resource Identifiers (IRIs)*, http://www.ietf.org/rfc/rfc3987.txt, RFC 3987, January 2005.
[RFC1737]	K. Sollins, L. Masinter, *Functional Requirements for Uniform Resource Names*, http://www.ietf.org/rfc/rfc1737.txt, RFC 1737, December 1994.
[RFC2119]	S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, RFC 2119, March 1997.
[RFC2141]	R. Moats, *URN Syntax*, http://www.ietf.org/rfc/rfc2141.txt, IETF RFC 2141, May 1997.
[RFC2234]	D. H. Crocker and P. Overell, *Augmented BNF for Syntax Specifications: ABNF*, http://www.ietf.org/rfc/rfc2234.txt, RFC 2234, November 1997.
[RFC2718]	L. Masinter, H. Alvestrand, D. Zigmond, R. Petke, *Guidelines for New URL Schemes*, http://www.ietf.org/rfc/rfc2718.txt, RFC 2718, November 1999.
[RFC2732]	R. Hinden, B. Carpenter, L. Masinter, *Format for Literal IPv6 Addresses in URL's*, http://www.ietf.org/rfc/rfc2732.txt, RFC 2732, December, 1999.
[RFC3305]	M. Mealing, R. Denenberg, *Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations*, http://www.ietf.org/rfc/rfc3305.txt, RFC 3305, August 2002.
[RFC3491]	P. Hoffman, M. Blanchet, *Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)*, http://www.ietf.org/rfc/rfc3491, RFC 3491, March 2003.
[RFC3629]	F. Yergeau, *UTF-8, A Transformation Format of ISO 10646*, http://www.faqs.org/rfcs/rfc3629.html, RFC 3629, November, 2003.
[UniXML]	M. Dürst, A. Freytag, *Unicode in XML and other Markup Languages*, Unicode Technical Report #20, World Wide Web Consortium Note, February 2002.
[URI]	T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifier (URI): Generic Syntax,* http://www.ietf.org/rfc/rfc3986.txt, STD 66, RFC 3986, January 2005.
[UTR15]	M. Davis, M. Dürst, *Unicode Normalization Forms*, http://www.unicode.org/unicode/reports/tr15/tr15-23.html, April 17, 2003.

4.2 Informative

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

Appendix A. Collected ABNF for XRI (Normative)

815 This section contains the complete ABNF for XRI syntax. XRI productions use green shading,
816 while productions inherited from IRI use yellow shading. A valid XRI MUST conform to this ABNF.
817

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

820   xri-hier-part     = ( xri-authority / iauthority ) xri-path-abempty

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

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

824   relative-XRI-ref  = relative-XRI-part [ "?" iquery ] [ "#" ifragment ]

825   relative-XRI-part = xri-path-absolute
826                       / xri-path-noscheme
827                       / ipath-empty

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

829   xri-no-scheme     = xri-hier-part [ "?" iquery ]
830                       [ "#" ifragment ]

831   xri-authority     = gcs-authority
832                       / xref-authority

833   gcs-authority     = pgcs-authority / rgcs-authority

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

835   rgcs-authority    = rgcs-char xri-segment

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

837   xref-authority    = xref *xri-subseg

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

839   xri-path          = xri-path-abempty
840                       / xri-path-absolute
841                       / xri-path-noscheme
842                       / ipath-empty

843   xri-path-abempty  = *( "/" xri-segment )

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

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

846   xri-segment       = xri-subseg-od *xri-subseg

847   xri-segment-nz    = xri-subseg-od-nz *xri-subseg
```

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

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

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

851    xri-subseg-od-nz  = [ "*" / "!" ] (xref / 1*xri-pchar)

852    xri-subseg-od-nx  = [ "*" / "!" ] 1*xri-pchar-nc

853    xri-subseg-pt-nz  = "!" (xref / 1*xri-pchar)

854    xri-pchar         = iunreserved / pct-encoded / xri-sub-delims / ":"

855    xri-pchar-nc      = iunreserved / pct-encoded / xri-sub-delims

856    xri-reserved      = xri-gen-delims / xri-sub-delims

857    xri-gen-delims    = ":" / "/" / "?" / "#" / "[" / "]" / "(" / ")"
858                      / "*" / "!" / rgcs-char

859    xri-sub-delims    = "&" / ";" / "," / "'"

860    IRI               = scheme ":" ihier-part [ "?" iquery ]
861                      [ "#" ifragment ]

862    scheme            = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )

863    ihier-part        = "//" iauthority ipath-abempty
864                      / ipath-abs
865                      / ipath-rootless
866                      / ipath-empty

867    iauthority        = [ iuserinfo "@" ] ihost [ ":" port ]

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

869    ihost             = IP-literal / IPv4address / ireg-name

870    IP-literal        = "[" ( IPv6address / IPvFuture  ) "]"

871    IPvFuture         = "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":" )

872    IPv6address       =                            6( h16 ":" ) ls32
873                      /                       "::" 5( h16 ":" ) ls32
874                      / [               h16 ] "::" 4( h16 ":" ) ls32
875                      / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
876                      / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
877                      / [ *3( h16 ":" ) h16 ] "::"    h16 ":"   ls32
878                      / [ *4( h16 ":" ) h16 ] "::"              ls32
879                      / [ *5( h16 ":" ) h16 ] "::"              h16
880                      / [ *6( h16 ":" ) h16 ] "::"

881    ls32              = ( h16 ":" h16 ) / IPv4address

882    h16               = 1*4HEXDIG

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

```
884    dec-octet         = DIGIT                ; 0-9
885                      / %x31-39 DIGIT        ; 10-99
886                      / "1" 2DIGIT           ; 100-199
887                      / "2" %x30-34 DIGIT    ; 200-249
888                      / "25" %x30-35         ; 250-255

889    ireg-name         = *( iunreserved / pct-encoded / sub-delims )

890    port              = *DIGIT

891    ipath-abempty     = *( "/" isegment )

892    ipath-abs         = "/" [ isegment-nz *( "/" isegment ) ]

893    ipath-rootless    = isegment-nz *( "/" isegment )

894    ipath-empty       = 0<ipchar>

895    isegment          = *ipchar

896    isegment-nz       = 1*ipchar

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

898    iprivate          = %xE000-F8FF / %xF0000-FFFFD / %x100000-10FFFD

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

900    ipchar            = iunreserved / pct-encoded / sub-delims / ":" / "@"

901    iunreserved       = ALPHA / DIGIT / "-" / "." / "_" / "~" / ucschar

902    pct-encoded       = "%" HEXDIG HEXDIG

903    ucschar           = %xA0-D7FF / %xF900-FDCF / %xFDF0-FFEF
904                      / %x10000-1FFFD / %x20000-2FFFD / %x30000-3FFFD
905                      / %x40000-4FFFD / %x50000-5FFFD / %x60000-6FFFD
906                      / %x70000-7FFFD / %x80000-8FFFD / %x90000-9FFFD
907                      / %xA0000-AFFFD / %xB0000-BFFFD / %xC0000-CFFFD
908                      / %xD0000-DFFFD / %xE1000-EFFFD

909    reserved          = gen-delims / sub-delims

910    gen-delims        = ":" / "/" / "?" / "#" / "[" / "]" / "@"

911    sub-delims        = "!" / "$" / "&" / "'" / "(" / ")"
912                      / "*" / "+" / "," / ";" / "="

913    unreserved        = ALPHA / DIGIT / "-" / "." / "_" / "~"
```

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

To leverage existing infrastructure, it may sometimes be useful to convert HTTP IRIs into XRIs. Because XRI syntax is, for the most part, a superset of generic IRI syntax, the majority of HTTP IRIs can be converted to valid XRIs simply by replacing the scheme name "http" with "xri". Generally the authority component of the resulting XRI will be properly interpreted as an IRI authority. There may be some cases, however, in which a legal authority component in an IRI will be interpreted as an XRI authority after this conversion. For example,

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

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

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

Because the authority segment "!!1" matches both the "xri-authority" and the "iauthority" ABNF productions, it would be interpreted as an XRI authority based on the "first-match-wins" rule used to resolve ambiguities in the ABNF. Section 2.2.1.2 provides other examples of legal IRI authorities that would be interpreted as XRI authorities when used in an XRI. However these cases are unlikely to arise in practice since they typically result in an invalid URI when converted from an IRI.

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

Appendix C. Glossary

The following definitions are used in specifications from the OASIS XRI Technical Committee
Note that this glossary supercedes the glossary in **[XRIReqs]**.

Absolute Identifier

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

Abstract Identifier

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

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

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

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

Authority (or Identifier Authority)

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

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

Base Identifier

An absolute identifier that identifies a context for a relative identifier. Changing the base
identifier changes the context of the relative identifier. See "Relative Identifier."

Canonical Form

The form of an identifier after applying transformation rules for the purpose of determining
equivalence. See also "Normal Form".

Community (or Identifier Community)

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

Concrete Identifier

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

Context (or Identifier Context)

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

Cross-reference

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

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

Delegated Identifier

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

Federated Identifier

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

Global Context Symbol (GCS)

A reserved character used at the start of the authority segment of an XRI to establish the global context of an XRI authority. See section 2.2.1.2.

Hierarchy

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

Human-Friendly Identifier (HFI)

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

Identifier

Per **[URI]**, anything that "embodies the information required to distinguish what is being identified from all other things within its scope of identification." In UML terms, an identifier is an attribute of a resource (the identifier context) that forms an association with another resource (the identifier target). The general term "identifier" does not specify whether the identifier is abstract or concrete, absolute or relative, persistent or

| 1031 | reassignable, human-friendly or machine-friendly, delegated or local, hierarchical or |
| 1032 | polyarchical, or resolvable or self-referential. |

I-name

1034   An informal term used to refer to a reassignable XRI; more specifically, an XRI in which
1035   at least one sub-segment is reassignable.

I-number

1037   An informal term used to refer to a persistent XRI; more specifically, an XRI in which all
1038   sub-segments are persistent. Note that a persistent XRI is not required to be numeric—it
1039   may be any text string meeting the XRI ABNF requirements.

IRI (Internationalized Resource Identifier)

1041   IRI is a specification for internationalized URIs developed by the W3C. IRIs specify how
1042   to include characters from the Universal Character Set (Unicode/ISO10646) in URIs. The
1043   IRI specification **[IRI]** provides a mapping from IRIs to URIs, which allows IRIs to be used
1044   instead of URIs where appropriate. This XRI specification defines a similar transformation
1045   from XRIs to IRIs for the same reason.

IRI Authority

1047   An identifier authority (see "Authority") represented by the authority segment of an XRI
1048   that does not match the "xri-authority" production but matches the "iauthority" production
1049   from **[IRI]**. See section 2.2.1.3. Mutually exclusive with "XRI Authority".

Local Identifier

1051   Any identifier, or any set of segments in a multi-segment identifier, that is assigned by the
1052   same identifier authority. Each of these segments is local to that authority. Mutually
1053   exclusive with "Delegated Identifier."

Machine-Friendly Identifier (MFI)

1055   An identifier containing digits, hexadecimal values, or other character sequences
1056   optimized for efficient machine indexing, searching, routing, caching, and resolvability.
1057   MFIs generally do not contain human semantics. Compare with "Human-Friendly
1058   Identifier."

Normal Form

1060   The character-by-character format of an identifier after encoding, escaping, or other
1061   character transformation rules have been applied in order to satisfy syntactic
1062   requirements. Three normal forms are defined for XRIs—XRI-normal form, IRI-normal
1063   form, and URI-normal form. See section 2.3.1 for details. See also "Canonical Form".

Path

1065   The relationships between resources defined by a multi-segment identifier. In less strict
1066   contexts, the word "path" often refers to the multi-segment identifier itself, or to the
1067   resources it represents (such as filesystem directories).

Persistent Identifier

1069   An identifier that is permanently assigned to a resource and intended never to be
1070   reassigned to another resource - even if the original resource goes off the network, is
1071   terminated, or ceases to exist. A URN as described in **[RFC2141]** is an example of a
1072   persistent identifier. Persistent identifiers tend to be machine-friendly identifiers, since
1073   human-friendly identifiers often reflect human semantic relationships that may change
1074   over time. Mutually exclusive with "Reassignable Identifier."

Polyarchy

1076   A treelike structure composed of multiple intersecting hierarchies in which primary
1077   relationships can cross hierarchies. A polyarchy allows one member to be connected or

1078 linked to any other. In contrast to a web, however, the overall structure tends to remain
1079 strongly hierarchical. XRIs support polyarchic paths through the use of cross-references.
1080 See also "Cross-reference" and "Hierarchy".

Reassignable Identifier

1081

1082 An identifier that may be reassigned from one resource to another. Example: the domain
1083 name "example.com" may be reassigned from ABC Company to XYZ Company, or the
1084 email address "mary@example.com" may be reassigned from Mary Smith to Mary Jones.
1085 Reassignable identifiers tend to be human-friendly because they often represent the
1086 potentially transitory mapping of human semantic relationships onto network resources or
1087 resource representations. Mutually exclusive with "Persistent Identifier."

Relative Identifier

1088

1089 An identifier that refers to a resource only in relationship to a particular context (for
1090 example, the current community, the current document, or the current position in a
1091 delegated identifier). If the context changes, the identifier's meaning also changes. A
1092 relative identifier can be converted into an absolute identifier by combining it with a base
1093 identifier (an absolute identifier that is used to identify a context). See "Base Identifier".
1094 Mutually exclusive with "Absolute Identifier."

Resolvable Identifier

1095

1096 An identifier that references a network resource or resource representation and that can
1097 be dereferenced using a resolution protocol or other mechanism into a network endpoint
1098 for communicating with the target resource. Mutually exclusive with "Self-Reference."

Resource

1099

1100 Per **[URI]**, "anything that can be named or described." Resources are of two types:
1101 network resources (those that are network-addressable) and non-network resources
1102 (those that exist entirely independent of a network). Network resources are themselves of
1103 two types: physical resources (resources physically attached to or operating on the
1104 network) or resource representations (see "Resource Representation").

Resource Representation

1105

1106 A network resource that represents the attributes of another resource. A resource
1107 representation may represent either another network resource (such as a machine,
1108 service, application, file, or digital object) or a non-network resource (such as a person,
1109 organization, or concept).

Segment (or Identifier Segment)

1110

1111 Any syntactically delimited component of an identifier. In generic URI syntax, all
1112 segments after the authority portion are delimited by forward slashes
1113 ("/segment1/segment2/…"). In XRI syntax, slash segments can be further subdivided into
1114 sub-segments called *star segments* (for reassignable identifiers) and *bang segments* (for
1115 persistent identifiers). See section 2.2.3. XRI also supports another type of segment
1116 called a cross-reference, which is enclosed in parentheses. See "Cross-Reference".

Self-Reference (or Self-Referential Identifier)

1117

1118 An identifier which is itself the representation of the resource it references. Self-
1119 references are typically used to represent non-network resources (e.g., "love", "Paris",
1120 "the planet Jupiter") in contexts where an identifier is not intended to be resolved to a
1121 separate network representation of that resource. The primary purpose of self-references
1122 is to establish equivalence across contexts (see "Cross-References"). Mutually exclusive
1123 with "Resolvable Identifier."

Sub-segment

1124

1125 A syntactically delimited component of an identifier segment (see "Segment"). While URI
1126 and IRI syntax define only segments, XRI syntax defines both segments and sub-

1127   segments. XRI sub-segments are used to distinguish between persistent identifiers,
1128   reassignable identifiers, and cross-references. See sections 2.2.2 and 2.2.3.

Synonym (or Identifier Synonym)

1129

1130   An identifier that is asserted by an identifier authority to be equivalent to another identifier
1131   not because of strict literal equivalence, but because it resolves to the same resource.

Target (or Identifier Target)

1132

1133   The resource referenced by an identifier. A target may be either a network resource
1134   (including a resource representation) or a non-network resource.

URI (Uniform Resource Identifier)

1135

1136   The standard identifier used in World Wide Web architecture. Starting in 1998, RFC 2396
1137   has been the authoritative specification for URI syntax. In January 2005 it was
1138   superseded by RFC 3986 **[URI]**.

XDI (XRI Data Interchange)

1139

1140   A generalized, extensible service for sharing, linking, and synchronizing XML data and
1141   metadata associated with XRI-identified resources. XDI is being developed by the OASIS
1142   XDI Technical Committee (http://www.oasis-open.org/committees/xdi).

XRI Authority

1143

1144   An identifier authority (see "Authority") represented by the authority segment of an XRI
1145   that begins with either a global context symbol or a cross-reference. See section 2.2.1.1.
1146   Mutually exclusive with "IRI Authority."

XRI Reference

1147

1148   A term that includes both absolute and relative XRIs. Used in the same way as "URI
1149   reference" and "IRI reference." Note that to transform an XRI reference into an XRI, it
1150   must first be converted to absolute form, which in the case of a relative XRI requires the
1151   use of a base XRI to establish context.

Appendix D. Acknowledgments

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

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

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

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

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

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

- RFC 1737
- RFC 2616
- RFC 2718
- RFC 3986 (STD 66) and its predecessor, RFC 2396
- RFC 3987
- XNS

Appendix E. Notices

1198

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.