# Extensible Resource Identifier (XRI) Generic Syntax and Resolution Specification

## Release Candidate 2, 20 November 2003

**Document identifier:**

wd-xri-specification-rc2

**Location:**

http://www.oasis-open.org/committees/xri

**Editors:**

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

**Abstract:**

This document is the normative technical specification for XRI generic syntax and resolution. For a non-normative introduction to the uses and features of XRIs, see the *XRI Primer*.

**Status:**

This document is a working draft updated periodically on no particular schedule. Send comments to the editors.

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

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

The errata page for this specification is at http://www.oasis-open.org/committees/xri/yyy.





# Table of Contents

# Introduction

## 1.1 Overview of XRIs

An Extensible Resource Identifier (XRI) provides 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.

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

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

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

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

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

### 1.1.1 Generic Syntax

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

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

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

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

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

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

### 1.1.2 Examples

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

```
xri://www.example.com/pages/index.html
       --standard HTTP URI converted to an XRI

xri://[2010:836B:4179::836B:4179]/pages/index.html
       --using an IPv6 authority per RFC 2732

xri://www.example.com/inventory.parts/widget.subwidget.foobarator
       --delegation of reassignable identifiers

xri://www.example.com/:inventory:parts/:12:7:234
       --delegation of persistent identifiers

xri:@ExampleCorp
xri:@ExampleCorp.www
xri:@ExampleCorp.website
xri:=JohnDoe
xri:=JohnDoe.home
xri:=JohnDoe.work
xri:+flowers
xri:+flowers.rose
xri:+flowers.daisy
       --global context symbols

xri://www.example.com/(+management)/(+CEO)
xri:(urn:oasis:spec:2040)/(+index)
xri:(mailto:john.doe@example.com)/(+phone)
xri:=JohnDoe.home/(+email)
xri:=JohnDoe.home/(+email).($v/3)
       --cross-references

xri:(+flowers.rose)
```

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

Table 1 also illustrates several examples of internationalized XRIs.

| | |
|---|---|
| French | xri:@ALaFrançaise/areté |
| Hebrew | xri://א.בג.ef/gh/יט/כל.html |
| Kanji | xri:=崎村夏彦/(+本籍地) |

Table 1: Internationalized XRIs.

### 1.1.3 URI, URL, URN, and XRI

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

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

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

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

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

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

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

## 1.2 Design Considerations

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

### 1.2.1 Abstraction and Independence

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

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

### 1.2.2 Persistence and Reassignability

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

### 1.2.3 Human-Friendliness and Machine-Friendliness

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

### 1.2.4 Internationalization

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

### 1.2.5 Cross-Context Identification

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

### 1.2.6 Authority, Delegation, and Federation

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

### 1.2.7 Security and Privacy

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

### 1.2.8 Extensibility

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

## 1.3 Terminology and Notation

### 1.3.1 Keywords

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

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

### 1.3.2 Syntax Notation

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

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

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

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

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

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

### 1.3.3 Glossary

The following definitions are central to this specification.

**Absolute Identifier**

An identifier that refers to a resource independent of the current context, i.e., using 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 another abstract identifier or a concrete identifier.)

A URN as described in **[RFC2141]** is an example of an abstract identifier. Abstract identifiers provide 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)**

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

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

**Base Identifier**

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

**Canonical Form**

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

**Community (or Identifier Community)**

The set of resources that share a common identifier authority, often (but not always) a common root authority. Technically, the 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, a phone number that rings directly to a specific device, and a postal address that is not a forwarding address. All concrete identifiers are intended to be resolvable identifiers. Contrast with “Abstract Identifier.”

**Context (or Identifier Context)**

The resource of which an identifier is an attribute. For example, in the string of delegated identifiers “a/b/c”, the context of the identifier “b” is “a/”, and the context of the identier “c” is “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. For relative identifiers, the context is implicit.

**Cross-reference**

An identifier assigned in one context that is reused in another context. Cross-references are used primarily to identify logically equivalent resources in different domains 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 page stored on multiple proxy servers, the same datatype used in multiple databases or XML schemas, or the same abstract concept used in multiple taxonomies or ontologies.

**Delegated Identifier**

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

**Federated Identifier**

A delegated identifier which spans independent identifier authorities. See also “Delegated Identifier.”

**Human-Friendly Identifier (HFI)**

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

**Identifier**

Per **[RFC2396bis]**, 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, persistent or reassignable, human-friendly or machine-friendly, absolute or relative, resolvable or self-referential, or delegated or local.

**Local Identifier**

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

**Machine-Friendly Identifier (MFI)**

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

**Normal Form**

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

**Persistent Identifier**

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

**Reassignable Identifier**

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

**Relative Identifier**

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

**Resolvable Identifier**

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

**Resource**

Per **[RFC2396bis]**, “anything that can be named or described.” Resources are of two types: network resources (those that are network addressable) and non-network resources (those that exist entirely independent of a network). Network resources may be either direct resources or resource representations (see “Resource Representation”).

**Resource Representation**

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

**Segment**

Any syntactically delimited portion of an identifier. In generic URI syntax, all segments after the authority portion are delimited by forward slashes ("/segment1/segment2/…"). In XRI syntax, slash segments can be further subdivided into sub-segments called *dot segments* (for reassignable identifiers) and *colon segments* (for persistent identifiers). See section 2.1.2.

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

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

**Target (or Identifier Target)**

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

**XRI Reference**

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

# 2 Syntax

## 2.1 Syntax Components

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

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

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

An absolute XRI consists of the scheme name “xri:” followed by the same set of hierarchical components as an absolute URI – authority, path, query, and fragment. For convenience in the ABNF, these components are broken into the *authority path*, the *local path*, and the *query-frag* (a query segment, a fragment segment, or both).

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

A relative XRI consists of the same set of components as a relative URI.

```
relative-xri    = ( local-path / relative-path ) [ query-frag ]
```

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

```
xri-value       = [ global-path / local-path / relative-path ]
                  [ query-frag ]
```

### 2.1.1 Authority

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

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

#### 2.1.1.1 URI Authority

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

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

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

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

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

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

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

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

```
hostname        = idomainlabel qualified
qualified       = *( "." idomainlabel ) [ "." ]
idomainlabel    = 1*ucschar
domainlabel     = alphanum [ 0*61( alphanum / "-" ) alphanum ]
alphanum        = ALPHA / DIGIT
```

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

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

```
IPv6reference   = "[" IPv6address "]"
IPv6address =   6( h4 ":" ) ls32
                / "::" 5( h4 ":" )                          ls32
                / [              h4 ] "::" 4( h4 ":" )      ls32
                / [ *1( h4 ":" ) h4 ] "::" 3( h4 ":" )      ls32
                / [ *2( h4 ":" ) h4 ] "::" 2( h4 ":" )      ls32
                / [ *3( h4 ":" ) h4 ] "::"    h4 ":"        ls32
                / [ *4( h4 ":" ) h4 ] "::"                  ls32
                / [ *5( h4 ":" ) h4 ] "::"                  h4
                / [ *6( h4 ":" ) h4 ] "::"
ls32            = ( h4 ":" h4 ) / IPv4address
                ; least-significant 32 bits of address
h4              = 1*4HEXDIG
```

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

```
port            = *DIGIT
```

### 2.1.1.2 XRI Authority

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

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

### 2.1.1.3 Global Context Symbols (GCS)

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

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

The global context symbol characters were selected from the set of symbol characters that are valid in a URI under **[RFC2396]** to represent the global contexts shown in Table 2:

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

Table 2: XRI global context symbols.

Note that because a global context symbol may precede an xri-segment, and an xri-segment may start with a cross-reference (below), a global context symbol can be used to express the abstract logical context of a conventional URI authority. For example:

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

### 2.1.1.4 Cross-References

Cross-references are the primary extensibility mechanism in XRI. A cross-reference may be either an XRI value or an absolute URI. In either case, it is enclosed in parentheses the same way an IPv6 literal is encapsulated in square brackets as specified in **[RFC2732]** (see section 2.1.1.1).

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

It is important that the value of a cross-reference be syntactically unambiguous, whether it is an absolute URI or one of the various forms of an XRI value. Since an absolute URI must start with a legal URI scheme name character (i.e., an ALPHA), an XRI value used as a cross-reference MUST start with a symbol character. Since the only XRI value that is not required to start with a symbol is a dot segment (see section 2.1.2), the effect of this rule is that if a relative XRI begins with a dot segment, the leading dot is not optional. For example, if the relative XRI “foo/bar” is used as a cross-reference, it must include the optional leading dot, i.e., “(.foo/bar)”.

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

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

### 2.1.1.5 Self-References

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

```
The term "user-friendly" is used frequently in computing.
       --English-language usage of a quoted term

xri:(+user-friendly)
       --XRI syntax for expressing a self-reference
```

### 2.1.2 Path

As with URIs, the XRI path component is a hierarchal sequence of path segments separated by slash (“/”) characters and terminated by the first question-mark (“?”) or number sign (“#”) character, or by the end of the XRI. The key difference is that while a URI path segment is considered opaque by a generic URI processor, an XRI path segment can be parsed by an XRI processor into two types of sub-segments: *dot segments* and *colon segments* after their leading characters (“.” and “:”).

```
xri-segments    = xri-segment *( "/" xri-segment )
xri-segment     = ( [ "." ] sub-segment / ":" sub-segment )
                *( "." sub-segment / ":" sub-segment )
sub-segment     = *xri-pchar / xref
```

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

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

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

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

### 2.1.3 Query

The XRI query component is identical to the URI query component as described in section 3.4 of **[RFC2396]**, except that it may begin with a cross-reference. This permits the incorporation of XRI metadata describing the query string syntax. See the XRI Metadata Specification [tktk need reference] for more about query syntax metadata.

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

The characters permitted in a query segment are the same ones allowed in a URI query segment.

```
pchar           = unreserved / escaped / ";" /
                ":" / "@" / "&" / "=" / "+" / "$" / ","
```

### 2.1.4 Fragment

XRI syntax also supports fragments as described in section 4.1 of **[RFC2396]**, except that an XRI fragment may begin with a cross-reference.

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

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

## 2.2 Characters

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

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

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

### 2.2.1 Character Encoding

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

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

### 2.2.2 Reserved Characters

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

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

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

### 2.2.3 Unreserved Characters

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

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

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

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

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

## 2.2.4 Escaped Characters

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

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

### 2.2.4.1 Escaped Encoding

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

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

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

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

### 2.2.4.2 Encoding XRI Metadata

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

Another case is when the normalization or canonicalization rules of a particular identifier authority do not permit the inclusion of whitespace, mixed case letters, or certain punctuation in an XRI segment even when escaped, yet the authority would like to retain this metadata for purposes of presentation. XRI syntax offers an option for encoding this metadata using a cross-reference

beginning with the GCS “$” symbol. As defined in section 2.1.1.3, the top level authority for these identifiers is the XRI Metadata Specification [tktk – need reference]. It defines special identifiers for UTF-8 metadata, presentation metadata, and other standard types of identifier metadata together with the rules governing their interpretation.

### 2.2.4.3 Transforming XRIs into IRIs and URIs

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

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

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

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

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

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

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

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

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

### 2.2.4.4 Special Escaping Rules for XRI Syntax

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

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

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

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

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

The desired separation is:

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

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

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

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

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

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

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

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

instead of the intended:

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

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

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

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

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

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

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

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

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

### 2.2.4.5 Transforming URIs and IRIs Back into XRIs

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

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

1. If the identifier is not encoded in US-ASCII, convert it to a sequence of octets in US-ASCII.
2. If the identifier has a “hostname” component, replace it with the UTF-8 encoded “hostname” component converted using the “ToUnicode” operation defined in section 4.2 of **[RFC3490]**, with the “UseSTD3ASCIIRules” flag set to true and the “AllowedUnassigned” flag set to false.
3. Convert all escaped characters (as defined in section 2.2.4) with their corresponding octets, except for the percent “%” character, those characters in the “reserved” production of **[RFC2396]** and US-ASCII characters disallowed in URIs by section 2.4.3 of **[RFC2396]**.

4. Re-escape any octet produced in step 3 that is not part of a strictly legal UTF-8 octet sequence.
5. Perform the following special conversions for XRI syntax:
    a. Convert all escaped slash “/” characters to their corresponding octets.
    b. Convert all escaped question mark “?” characters to their corresponding octets.
    c. Convert all escaped number sign “#” characters to their corresponding octets.
    d. Convert all escaped percent “%” characters to their corresponding octets.
6. Encode the resulting sequence in UTF-8 (except for that portion already converted by step 3).

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

1. If the XRI is not encoded in UTF-8, convert the XRI to a sequence of characters encoded in UTF-8, normalized according to Normalization Form C (NFC) as defined in **[UTR15]**.
2. Perform Step 2 above.
3. Perform Step 5 above.

If the XRI is in IRI normal form, perform the same steps as with an XRI in anyURI normal form, except skip step 2.

### 2.2.5 Excluded Characters

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

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

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

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

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

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

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

## 2.3 Relative XRI References

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

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

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

### 2.3.1 Establishing a Base XRI

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

### 2.3.2 Obtaining the Referenced XRI

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

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

```
if authority is defined then
   append "//" to result
   append authority to result
```

is replaced by

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

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

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

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

the following relative XRIs would be resolved as shown:

```
.g:h        =  xri:@a.b.c/d.e/.g:h
./g:h       =  xri:@a.b.c/d.e/g:h
g:h         =  g:h      (see section 2.3.3 below)
g           =  xri:@a.b.c/d.e/g
./g         =  xri:@a.b.c/d.e/g
g/          =  xri:@a.b.c/d.e/g/
/g          =  xri:@a.b.c/g
?y          =  xri:@a.b.c/d.e/?y
g?y         =  xri:@a.b.c/d.e/g?y
#s          =  (current document)#s
g#s         =  xri:@a.b.c/d.e/g#s
g?y#s       =  xri:@a.b.c/d.e/g?y#s
;x          =  xri:@a.b.c/d.e/;x
g;x         =  xri:@a.b.c/d.e/g;x
g;x?y#s     =  xri:@a.b.c/d.e/g;x?y#s
.           =  xri:@a.b.c/d.e/
./          =  xri:@a.b.c/d.e/
..          =  xri:@a.b.c/
../         =  xri:@a.b.c/
../g        =  xri:@a.b.c/g
```

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

```
../..       =  xri:@a.b.c/../
../../      =  xri:@a.b.c/../
../../g     =  xri:@a.b.c/../g
```

### 2.3.3 Leading Segments Containing a Colon

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

> Authors should be aware that a path segment which contains a colon character cannot be used as the first segment of a relative URI path (e.g., “this:that”), because it would be mistaken for a scheme name.
>
> It is therefore necessary to precede such segments with other segments (e.g., “./this:that”) in order for them to be referenced as a relative path.

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

## 2.4 Normalization and Comparison

In general, the normalization and comparison rules for generic URIs specified in **[RFC2396]** apply to XRIs in URI normal form, namely that the scheme and hostname are case insensitive. This

section describes a number of additional XRI-specific rules for normalization and comparison. To reduce the requirements imposed upon a minimally conforming processor, the majority of these rules are RECOMMENDED rather than REQUIRED. An implementation that fails to observe them, however, may frequently treat two XRIs as non-equal when in fact they are equal. In addition to these rules, Section 6 of **[RFC2396bis]** offers advice on more aggressive strategies for normalization. Although entirely non-normative, implementers may find this information useful in developing a strategy for establishing equivalence, particularly with respect to XRIs containing cross-references to URIs.

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

### 2.4.1 Case

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

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

### 2.4 Encoding, Escaping, and Transformations

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

### 2.4.3 Optional Syntax

- An xri-segment (section 2.1.2) that omits the optional leading dot (“.”)is equivalent to the same xri-segment prefixed with the leading dot. For example the segment “/foo.bar” is equivalent to the segment “/.foo.bar”.
- 2.10A cross-reference (section 2.1.1.4) that begins with the GCS symbol for annotations (“!”) AND the delimiter that precedes the cross-reference SHOULD be ignored entirely for purposes of comparision. For example, “xri:@:A6B4.(!www.example.org):5E32” is equivalent to “xri:@:A6B4:5E32”. Note that because XRI annotations are explicitly designed to be ignored by XRI processors, failure to observe this rule will cause XRIs that are intended to be equivalent to be incorrectly evaluated.





### 2.4.4 Cross-References

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

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

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

### 2.4.5 Canonicalization

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

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

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

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

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

| | | |
|---|---|---|
| xri:@example/./abc | xri:@example/abc | Avoid /./ and /../ in absolute XRIs |

1132 Table 3: Examples of XRI canonicalizaton recommendations.

# 3 Resolution

## 3.1 Introduction

XRI resolution is the process of dereferencing an XRI to a network endpoint in order to communicate with the resource identified by the XRI. Because XRIs may be used across a wide variety of communities and applications, including as database keys, filenames, directory keys, object IDs, and XML IDs, no single resolution mechanism may be appropriate for all XRIs. However, in the interest of promoting interoperability, this specification defines a simple, flexible resolution protocol that relies exclusively on HTTP (or HTTPS) for network transport.

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

### 3.1.1 Assumptions

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

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

### 3.1.2 Phases of Resolution

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

- Authority resolution
- Local access

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

Figure 1 illustrates these two phases of XRI resolution:

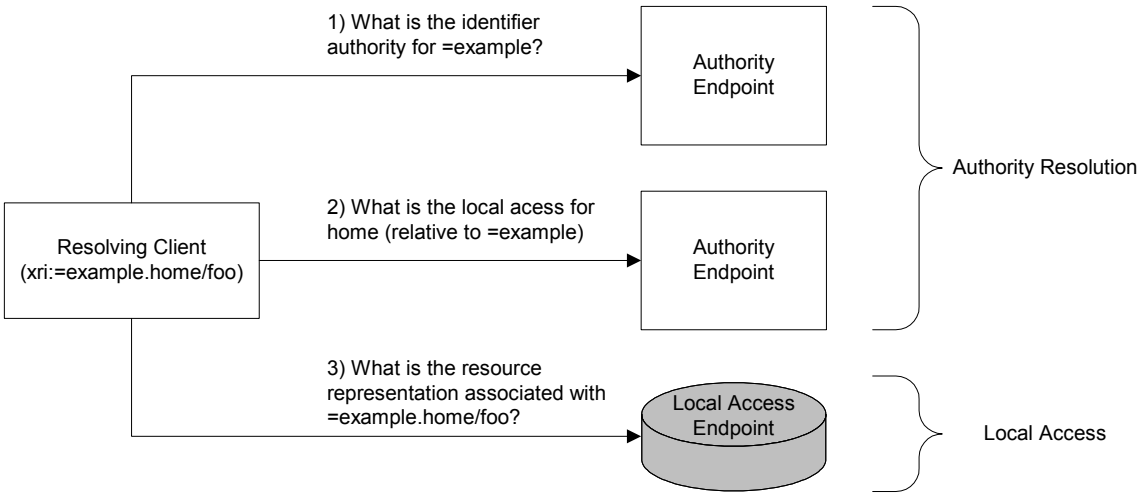


Figure 1: Phases of Resolution

### 3.1.3 XRI vs. URI Authorities

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

### 3.1.4 XRI Metadata Reserved for XRI Resolution

As defined in section 2.1.1.3, the GCS symbol "$" is reserved for XRI metadata, i.e., special identifiers assigned by this specification or the XRI Metadata Specification to describe or resolve other identifiers.

Within the "$" namespace, the identifier "$r" is reserved for identifiers assigned by this resolution specification. Table 4 summarizes these identifiers.

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

Table 4: Special identifiers reserved for XRI resolution.

## 3.2 XRI Authority Resolution

### 3.2.1 Overview

XRI authority resolution is an iterative process that resolves the sub-segments within the XRI authority segment from left to right. Each sub-segment is resolved in the context of the sub-segment immediately to the left. The first (or leftmost) sub-segment specifies the root of the identifier community. Each XRI community provides, by definition, one or more network endpoints

(HTTP or HTTPS URIs) that answer resolution requests for identifiers in its context. This starting point is further discussed in section 3.2.3.

After the first HTTP/HTTPS URI is determined based on the identifier community, the resolution process proceeds to the next sub-segment to the right. Each sub-segment is resolved to an Identifier Authority Descriptor as defined in section 3.2.2. This XML instance provides the data and metadata necessary to construct the URI for the next identifier authority as described in section 3.2.4. Once the final authority is reached, the Identifier Authority Descriptor provides the available local access service protocol(s) as discussed in section 3.4. In addition, the Identifier Authority Descriptor can provide a mapping of other XRIs used for this identifier authority.

All three options—next authority, local access, or mapping—are available at every iteration. For example, the XRI authority identifier “@a.b.c” may be the prefix to another identifier authority with the XRI “@a.b.c.d”, in which case “@a.b.c” must resolve to a Identifier Authority Descriptor. Or “@a.b.c” may be a local access endpoint itself, in which case its Identifier Authority Descriptor will contain references to local access services. Finally, this Identifier Authority Descriptor can also assert that the identifier “xri:@a.b.c” maps to the identifier “xri:@:1:2:3” in order to provide resolvers or caches with a long-term persistent XRI.

### 3.2.2 Identifier Authority Descriptors

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

The formal XML Schema definition of an XDI Identifier Authority Descriptor is provided in Appendix B. The following example illustrates the fields defined in this schema:

```
<IdentifierAuthority xmlns=”xri:$r.s/IdentifierAuthority”>
    <Resolved>.foo</Resolved>
    <NextAuthority>
        <URI>http://xri.example.com</URI>
        <URI>https://xri.example.com</URI>
    </NextAuthority>
    <LocalAccess>
        <Service> xri:$r.a/X2R</Service>
        <Type>application/rddl+xml</Type>
        <URI>http://xri.example.com</URI>
    </LocalAccess>
    <LocalAccess>
        <Service> xri:$r.a/X2R</Service>
        <Type>image/jpeg</Type>
        <URI>http://pictures.xri.example.com</URI>
    </LocalAccess>
    <Mapping>xri:@:1:2:3</Mapping>
</IdentifierAuthority>
```

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

**/IdentifierAuthority**

Required. The outer element of the IdentifierAuthority document.

**/IdentifierAuthority/Expires**

0 or 1. The UTC time at which this document MUST no longer be relied upon. A resolver MAY discard this document before the time indicated in this result. If the HTTP transport caching semantics specify an expiry time which is earlier than the time expressed in this

attribute, then the “IdentifierAuthority” document MUST no longer be relied upon after the expiry time declared in the HTTP headers per section 13.2 of **[RFC2616]**.

**/IdentifierAuthority/Resolved**

0 or 1. Expresses the XRI identifier sub-segment whose resolution results in this IdentifierAuthority document. This field can be used in conjunction with Digital Signatures to provide secure resolution (functionality that is not specified in this document, but which will be part of a future deliverable of the OASIS XRI TC). This field may also be useful for debugging or auditing purposes.

**/IdentifierAuthority/NextAuthority**

0 or 1. Indicates the next identifier authority to query if the just-resolved sub-segment represents another identifier authority. If the just-resolved sub-segment does not identify another identifier authority, but rather a resource in the context of the current identifier authority, then the “NextAuthority” element may not be present.

**/IdentifierAuthority/NextAuthority/URI**

1 or more. Required if “NextAuthority” element is present. Indicates the transport level URI where the next identifier authority can be contacted. Required to be an HTTP or HTTPS URI by this specification. Future extensions may use other transport protocols.

**/IdentifierAuthority/LocalAccess**

0 or more. Indicates that the just-resolved sub-segment specifies a identifier authority where local access service is available.

**/IdentifierAuthority/LocalAccess/Service**

0 or 1. Indicates the type of local access service. The service type is specified by a URI (including the URI normal form of an XRI). This specification defines one service: “X2R” which is identified with the URI “xri:$r.a/X2R” (see section 3.4.1.) An X2R service converts the XRI into a digital representation of that resource. No more specific semantics are defined. If this element is absent, then the service associated with this local access endpoint is X2R.

**/IdentifierAuthority/LocalAccess/Type**

0 or more. The media type of content available at this service. If this element is not present, then no assumption can be made about the type of data available at this endpoint. The content of this attribute must be of the form of a media type as defined in **[RFC2046]**. This element may appear multiple times to indicate multiple media types available through this local access service.

**/IdentifierAuthority/LocalAccess/URI**

1 or more. Required if “LocalAccess” element is present. Indicates the transport-level URI at which the local access service can be requested.

**/IdentifierAuthority/Mapping** and **/IdentifierAuthority/NextAuthority/Mapping**

0 or more. Represents another XRI that the Identifier authority specified by the resolution step may also be known as. This must be an absolute XRI (“absolute-xri” in the ABNF, section 2.1.)

Identifier authority mapping may be used, for example, to assert that a identifier authority known by a reassignable XRI may also be known by one or more persistent XRIs. Mapping may also be used to express that a particular authority may be known by a different reassignable XRI than the one which is being resolved. Both cases may be particularly useful in populating or querying a cache.

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

Another possible extension is the attachment of XML Digital Signatures and SAML assertions to support secure resolution. The current protocol does not specify such security features, although a secure resolution protocol is a future deliverable of the OASIS XRI TC and can be accommodated within the current Identifier Authority Descriptor schema.

### 3.2.3 Initiating Resolution

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

It is important to note that global context symbols are considered a segment by themselves. Therefore, if the sub-segment following the GCS does not begin with a colon (meaning it is not a persistent identifier), then a dot is implied. Table 5 and Table 6 demonstrate the parsing of such a sub-segment in the case of a GCS and a cross-reference, respectively.

| **XRI** | xri:@example.internal/foo |
|---|---|
| **Identifier Authority** | @example.internal |
| **Identifier Community** | @ |
| **First Sub-segment Resolved** | .example |

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

| **XRI** | xri:(http://www.example.com).internal/foo |
|---|---|
| **Identifier Authority** | (http://www.example.com).internal |
| **Identifier Community** | (http://www.example.com) |
| **First Sub-segment Resolved** | .internal |

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

### 3.2.4 Iterating Resolution

Once the Identifier Authority Descriptor representing the community root authority is known, the resolution process begins an iteration by constructing the Next Resolution URI. With each iteration, the Next Resolution URI is constructed by concatenating the same three parts:

1. The Next Authority URI extracted from the Identifier Authority Descriptor corresponding to the current context,
2. The XRI syntax delimiter (“.” or “:”) preceding the next sub-segment, and
3. The next sub-segment value itself (see the clarification regarding cross-references in section 3.2.6).

The URI which forms the base of the Next Resolution URI is the value of a URI element found at XPath **/IdentifierAuthority/NextAuthority/URI** in the Identifier Authority Descriptor. If this URI does not end with a “/” character, one must be appended before proceeding. Then this URI is

concatenated with the separator and the URI normal form (section 2.2.4.3) of the XRI sub-segment being resolved. As noted above, if there is no separator character preceding the sub-segment, a "." MUST be used for the separator.

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

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

```
ia-uri = authority-uri

if (authority-uri doesn't end in "/"):
    ia-uri = ia-uri + "/"

if (current-sub-segment isn't preceded with "." or ":" separator):
    ia-uri = ia-uri + "."
else:
     ia-uri  = ia-uri + separator

ia-uri = ia-uri + uri-escape(sub-segment)
```

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

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

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

### 3.2.5 Examples

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

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

Assume that the URI for the "=" global context symbol is "http://equals.xri.oasis-open.org/xri-resolve" (found in **/IdentifierAuthority/NextAuthority/URI** of the Identifier Authority Descriptor for this community). As explained in 3.2.3, this information, which provides a starting point for resolution, is known *a priori* and is part of the configuration of the resolver.

**Resolving "=example"**

The following HTTP request is made to "equals.xri.oasis-open.org":

```
GET /xri-resolve/.example HTTP/1.1
If-Modified-Since: Fri, 31 Oct 2003 19:43:31 GMT
```

```
<other HTTP headers>
```

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

```
200 OK HTTP/1.1
Content-Type: application/xriad+xml
Expires: Fri, 7 Nov 2003 19:43:31 GMT
<other HTTP headers>

<IdentifierAuthority xmlns=”…”>
<Resolved>.example</Resolved>
<NextAuthority>
<URI>
http://xri.example.com/xri-resolve/
</URI>
</NextAuthority>
<LocalAccess>…</LocalAccess>

</IdentifierAuthority>
```

**Resolving “=example.home”**

Appending the next sub-segment “home” to the URI “http://xri.example.com/xri-resolve/” yields the URI “http://xri.example.com/xri-resolve/.home”, and the following HTTP request is made to xri.example.com:

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

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

```
200 OK HTTP/1.1
Content-Type: application/xriad+xml
If-Modified-Since: Fri, 31 Oct 2003 19:43:32 GMT
<other HTTP headers>

<IdentifierAuthority xmlns=”…”>
<Resolved>.home</Resolved>
<NextAuthority>
<URI>
http://xri.home.example.com/xri-resolve/
</URI>
</NextAuthority>
<LocalAccess>…</LocalAccess>
…
</IdentifierAuthority>
```

**Resolving “=example.home.base”**

Appending the next sub-segment “base” to the URI “http://xri.home.example.com/xri-resolve/” gives the URI “http://xri. home.example.com/xri-resolve/.base”:

```
GET /xri-resolve/.base HTTP/1.1
If-Modified-Since: Fri, 31 Oct 2003 19:43:32 GMT

<other HTTP headers>
```

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

```
200 OK HTTP/1.1
Content-type: application/xriad+xml
Expires: Fri, 7 Nov 2003 19:43:33 GMT

<other HTTP headers>

<IdentifierAuthority xmlns="…">
<Resolved>.base</Resolved>
<NextAuthority>…</NextAuthority>
<LocalAccess>
<URI>
http://xrilocal.home.example.com/xri-local/
</URI>
<URI>
https://xrilocal.home.example.com/xri-local/
</URI>
</LocalAccess>
…
</IdentifierAuthority>
```

The result of the final Identifier authority resolution step is the set of HTTP and HTTPS URIs shown above that can be used for local access services.

### 3.2.6 Resolving Cross-References in XRI Authorities

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

Table 7 provides several examples. In each of these examples, sub-segment "b" resolves to a Next Authority URI of "http://example.com/xri-authority/".

| Cross-reference type | Example XRI | Next Resoluton URI after resolving "xri:@:a:b" |
|---|---|---|
| Absolute XRI | xri:@:a:b:(@:1:2:3).e/f | http://example.com/xri-authority/:(@:1:2:3) |
| Absolute URI | xri:@:a:b.(mailto:jd@example.com).e/f | http://example.com/xri-authority/.(mailto:jd@example.com) |
| Relative XRI | xri:@:a:b:(.c.d).e/f | http://example.com/xri-authority/:(.c.d) |

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

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

### 3.2.7 User Relative XRIs

A special case of XRI authority resolution is the user-relative context symbol ("*"). This symbol means the XRI authority is defined by the user of the XRI rather than being specified in the XRI itself. For example, a frequent XRI user could use the "*" symbol to enter their own XRI "shortcuts" or "speed names" that will be resolved into absolute XRIs using a mapping stored at the user's preferred identifier authority. Therefore, these XRIs are not resolvable without the *a*

*priori* establishment of an authority designated by the user, typically in the configuration of the user's XRI resolver.

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

Note that in most cases, mapping requires simply replacing the "*" character with a prefix corresponding to an absolute identifier authority. For example, if XRI resolver is pre-configured with a default community of "@employer/", then the XRI "xri:*Mary/workstation" would be converted into "xri:@employer/Mary/workstation".

## 3.3 URI Authority Resolution

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

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

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

## 3.4 Local Access

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

### 3.4.1 Local Access Service Types

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

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

### 3.4.2 HTTP/HTTPS Local Access

The HTTP/HTTPS local access protocol does not specify the semantics of the local access interaction, nor the form of the local access requests. The only semantics defined are those in **[RFC2616]**. Special attention should be paid to the semantics of the four main HTTP verbs: GET, PUT, POST, and DELETE. For example, clients performing local access typically would use GET when wishing to retrieve representations of a resource on the network.

This specification does not impose particular semantics beyond what is defined in **[RFC2616]**, but users of this specification are encouraged to review the **[REST]** architecture when building applications using XRIs. Local access is not limited to the REST model of interaction, however.

For example, HTTP local access could be leveraged for the delivery of SOAP messages over HTTP POST, or via use of the GET HTTP verb as a generic read-only resolution infrastructure.

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

### 3.4.3 Constructing a Local Access HTTP/HTTPS URI

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

The HTTP/HTTPS URI with which to perform local access is constructed by concatenating the Local Access URI from the Identifier Authority Descriptor (section 3.2.2) with the local part of the XRI. Specifically, the URI from the element identified with the relative XPATH **/IdentifierAuthority/LocalAccess/URI** is concatenated with the URI normal form (section 2.2.4.3) version of the remaining relative-path (section 2.1). If the LocalAccess URI does not terminate in a "/", it MUST be inserted before the relative-path.

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

```
concrete-http-uri = localaccess-uri

if (localaccess-uri does not end in "/"):
   concrete-http-uri = localaccess-uri + "/"

concrete-http-uri = localaccess-uri + uri-escape(relative-path)
```

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

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

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

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

```
200 OK HTTP/1.1
Expires: Sat, 1 Nov 2003 19:43:33 GMT
Content-Type: text/plain
<other HTTP headers>

This is the result of a local access request.
```

### 3.4.4 Using a Cross-Reference to Specify a Representation Type

A cross-reference MAY be used to specify a desired resource representation type when performing local access. The namespace "$r.t" is reserved for this purpose. This specification

does not enumerate such types; they are further defined in the XRI Metadata Specification [tktk need reference].

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

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

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

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

The resulting HTTP response would be:

```
200 OK HTTP/1.1
<cache-headers>
<other HTTP headers>

<content of representation of RDDL for xri:=example.home/foo.bar>
```

## 3.5 HTTP Headers

### 3.5.1 Caching

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

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

### 3.5.2 Location

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

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

### 3.5.3 Content-Location

“Content-Location” may be used during local access where the resource being accessed is an “attribute” or “view” of another resource. This usually would occur in the case where metadata is being accessed using a trailing cross reference to an XRI value under the “$r.t” namespace (see

section 3.4.4). Such a “Content-Location” header would specify where the resource itself may be accessible (rather than the metadata). This is not required and MUST NOT be required by resolving clients for proper operation. The content-location SHOULD be an HTTP/HTTPS URI if the local access server is aware of the HTTP/HTTPS location, otherwise it MAY be an XRI.

### 3.5.4 Content-Type

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

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

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

### 3.5.5 X-XRI-Canonical

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

Form:

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

## 3.6 Other HTTP Features

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

## 3.7 Caching and Efficiency

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

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

For Identifier Authority Descriptors, the cache expiration time may also be shortened by the expiration time provided in the Identifier Authority Descriptor at **/IdentifierAuthority/Expires** (if present). That is, if the expiration time in **/IdentifierAuthority/Expires** is sooner than the expiration time calculated from the HTTP caching semantics, then the Identifier Authority Descriptor SHOULD be discarded before the expiration time in **/IdentifierAuthority/Expires**.

With both application-level and HTTP-level caching, the resolution process is designed to have minimal overhead. In particular, because each sub-segment of an abstractly-identified XRI authority is resolved separately, each step of that resolution is a completely independent,

cacheable HTTP request. For this reason, resolution of top-level (leftmost) sub-segments, which are common to more identifiers, will naturally result in a greater number of cache hits than resolution of sub-segments further to the right.

## 3.8 Points of Extensibility

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

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

# 4 Security and Data Protection

## 4.1 Secure Resolution

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

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

## 4.2 XRI Metadata

The use of "$" cross-references for encoding XRI metadata in an XRI may involve other security and data protection considerations that are outside the scope of this specification. These considerations are addressed in the XRI Metadata Specification [tktk – need reference].

## 4.3 XRI Usage in Legacy Infrastructure

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

- Reliability and Consistency
- Malicious Construction
- Rare IP Address Formats
- Sensitive Information
- Semantic Attacks

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

## 4.4 XRI Usage in Evolving Infrastructure

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

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

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

# 5 References

## 5.1 Normative


**[RFC1737]** K. Sollins, L. Masinter, *Functional Requirements for Uniform Resource Names*, http://www.ietf.org/rfc/rfc1737.txt, RFC 1737, December 1994.

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

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

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

**[RFC2234]** D. H. Crocker and P. Overell, *Augmented BNF for Syntax Specifications: ABNF*, http://www.ietf.org/rfc/rfc2234.txt, RFC 2234, November 1997.

**[RFC2396]** T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifiers (URI): Generic Syntax,* http://www.ietf.org/rfc/rfc2396.txt, RFC 2396, August 1998.

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

**[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.

**[RFC3066]** H. Alvestrand, *Tags for the Identification of Languages,* http://www.ietf.org/rfc/rfc3066.txt, RFC 3066, January, 2001.

**[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.

**[RFC3490]** P. Faltstrom, P. Hoffman, A. Costello, *Internationalizing Domain Names in Applications (IDNA)*, http://www.ietf.org/rfc/rfc3490, RFC 3490, March 2003.

**[RFC3491]** P. Hoffman, M. Blanchet, *Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN),* http://www.ietf.org/rfc/rfc3491, RFC 3491, March 2003.

**[UML]** Object Management Group, *Unified Modeling Language (UML) Version 1.5*, http://www.omg.org/technology/documents/formal/uml.htm, March 1, 2003.

**[Unicode]** The Unicode Consortium. The Unicode Standard, Version 4.0.0, defined by: *The Unicode Standard, Version 4.0* (Boston, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1)

**[UniXML]** Duerst, M. and A. Freytag, *Unicode in XML and other Markup Languages*, Unicode Technical Report #20, World Wide Web Consortium Note, February 2002.

**[UTR15]** M. Davis, M. Duerst, *Unicode Normalization Forms*, http://www.unicode.org/unicode/reports/tr15/tr15-23.html, April 17, 2003.

**[XML]** T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, *Extensible Markup Language (XML) 1.0 (Second Edition) W3C Recommendation*, http://www.w3.org/TR/REC-xml, October 2000.

**[XMLSchema2]** P. Biron, A. Malhotra, *XML Schema Part 2: Datatypes W3C Recommendation,* http://www.w3.org/TR/xmlschema-2/, May 2001.

## 5.2 Informative

**[IRI]** M. Duerst, M. Suignard, *Internationalized Resource Identifiers (IRIs)*, http://www.ietf.org/internet-drafts/draft-duerst-iri-05.txt, Work-In-Progress, October 2003.

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

**[RFC2396bis]** R. Fielding, *Uniform Resource Identifiers (URI): Generic Syntax*, Internet Draft draft-fielding-uri-rfc2396bis-03, http://www.apache.org/~fielding/uri/rev-2002/rfc2396bis.html, Work-In-Progress, June 2003.

**[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)

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

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

absolute-xri    = "xri:" global-path

alphanum        = ALPHA / DIGIT

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

authority-path  = URI-authority / XRI-authority

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

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

domainlabel     = alphanum [ 0*61( alphanum / "-" ) alphanum ]

escaped         = "%" HEXDIG HEXDIG

excluded        = invisible / delims / unwise

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

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

global-path     = authority-path [ local-path ] [ query-frag ]

global-xri      = global-path [ "?" xri-query ] [ "#" xri-fragment ]

h4              = 1*4HEXDIG

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

host            = [ hostname / IPv4address / IPv6reference ]

hostname        = idomainlabel qualified

idomainlabel    = 1*ucschar

invisible       = CTL / SP / %x80-FF

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

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

```
                / [ *3( h4 ":" ) h4 ] "::"    h4 ":"       ls32
                / [ *4( h4 ":" ) h4 ] "::"                 ls32
                / [ *5( h4 ":" ) h4 ] "::"                 h4
                / [ *6( h4 ":" ) h4 ] "::"

IPv6reference   = "[" IPv6address "]"

local-path      = "/" relative-path

ls32            = ( h4 ":" h4 ) / IPv4address
                ; least-significant 32 bits of address

mark            = "-" / "_" / "." / "!" / "~" / "*" / "'" / "(" / ")"

net-path        = "//" authority [ abs-path ]

path-segments   = segment *( "/" segment )

pchar           = unreserved / escaped / ";" /
                ":" / "@" / "&" / "=" / "+" / "$" / ","

port            = *DIGIT

qualified       = *( "." idomainlabel ) [ "." ]

query           = *( pchar / "/" / "?" )

query-frag      = [ "?" xri-query ] [ "#" xri-fragment ]

relative-path   = *( [ "." ] "./" ) xri-segments

relative-xri    = ( local-path / relative-path ) [ query-frag ]

rel-path        = path-segments

reserved        = "/" / "?" / "#" / "[" / "]" / ";" /
                ":" / "@" / "&" / "=" / "+" / "$" / ","

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

segment         = *pchar

sub-segment     = *xri-pchar / xref

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

unreserved      = ALPHA / DIGIT / mark

unwise          = "{" / "}" / "|" / "\" / "^" / "`"

URI             = scheme ":" hier-part [ "?" query ] [ "#" fragment ]

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

uric            = reserved / unreserved / escaped

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

```
xref            = "(" ( xri-value / URI ) ")"

xref-authority  = xref  ( "." sub-segment / ":" sub-segment) *( "."
                sub-segment / ":" sub-segment)

XRI             = absolute-xri / relative-xri

XRI-authority   = ( gcs-char xri-segment ) / xref-authority

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

xri-fragment    = [ xref ] * ( pchar / "/" / "?" )

xri-mark        = "-" / "_" / "~" / "'"

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

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

xri-query       = [ xref ] * ( pchar / "/" / "?" )

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

xri-segment     = ( [ "." ] sub-segment / ":" sub-segment )
                *( "." sub-segment / ":" sub-segment )

xri-segments    = xri-segment *( "/" xri-segment )

xri-unreserved  = ALPHA / DIGIT / ucschar / xri-mark

xri-value       = [ global-path / local-path / relative-path ]
                [ query-frag ]
```

# Appendix B. XML Schema for XRI Identifier Authority Descriptor (Normative)

```
<xs:schema targetNamespace="xri:$r.s/IdentifierAuthority" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="xri:$r.s/IdentifierAuthority" elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:complexType name="IdentifierAuthorityType">
      <xs:sequence>
         <xs:element name="Resolved" type="ResolvedType" minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="Expires" type="ExpiresType" minOccurs="0"/>
         <xs:element name="NextAuthority" type="NextAuthorityType" minOccurs="0"/>
         <xs:element name="LocalAccess" type="LocalAccessType" minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="Mapping" type="MappingType" minOccurs="0" maxOccurs="unbounded"/>
         <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
   </xs:complexType>
   <xs:complexType name="NextAuthorityType">
      <xs:sequence>
         <xs:element name="URI" type="URIType" maxOccurs="unbounded"/>
         <xs:element name="Mapping" type="MappingType" minOccurs="0" maxOccurs="unbounded"/>
         <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
   </xs:complexType>
   <xs:complexType name="LocalAccessType">
      <xs:sequence>
         <xs:element name="Service" type="ServiceType" minOccurs="0"/>
         <xs:element name="Type" type="TypeType" minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="URI" type="URIType" maxOccurs="unbounded"/>
         <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
   </xs:complexType>
   <xs:element name="IdentifierAuthority" type="IdentifierAuthorityType"/>
   <xs:complexType name="ResolvedType">
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:anyAttribute namespace="##other" processContents="lax"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <xs:complexType name="URIType">
      <xs:simpleContent>
         <xs:extension base="xs:anyURI">
            <xs:anyAttribute namespace="##other" processContents="lax"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <xs:complexType name="ExpiresType">
      <xs:simpleContent>
         <xs:extension base="xs:dateTime">
            <xs:anyAttribute namespace="##other" processContents="lax"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <xs:complexType name="ServiceType">
      <xs:simpleContent>
         <xs:extension base="xs:anyURI">
            <xs:anyAttribute namespace="##other" processContents="lax"/>
         </xs:extension>
      </xs:simpleContent>
```

```
    </xs:complexType>
    <xs:complexType name="TypeType">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="MappingType">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:schema>
```

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

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

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

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

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

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

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

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

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

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

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

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

When converted to an XRI, as

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

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

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

could be similarly misinterpreted.

# Appendix D. Acknowledgments

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

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

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 2396 (and RFC 2396bis)
- RFC 2616
- RFC 2718
- RFC 3401-3405 (DDDS)
- REST Architecture
- IRI – Internationalized Resource Identifiers draft
- XNS

# Appendix E. Revision History

*[This appendix should be removed for specifications that are at OASIS Standard level.]*

| Rev | Date | By Whom | What |
|---|---|---|---|
| wd-01 | 2003-06-24 | Drummond Reed | Initial version to review structure and section 1 with other editors |
| wd-02 | 2003-06-25 | Drummond Reed | Reorganized overall structure and drafted first portion of section 2 |
| wd-03 | 2003-06-30 | Drummond Reed | Reorganized level two headings; edited section 1; drafted all ABNF portions of section 2; added collected ABNF to Appendix A; added Appendix B with initial $ identifiers; added Appendix C. |
| wd-04 | 2003-07-02 | Dave McAlpin | Editorial changes; new text in 2.2.3.2, 2.4, 2.4.*, 2.5, 2.5.*, 4.* |
| wd-05 | 2003-07-03 | Dave McAlpin | Editorial changes; added resolution text (section 3) |
| wd-06 | 2003-07-03 | Dave McAlpin | Minor edits; removed inline notes and created issues section as Appendix G. |
| wd-07 | 2003-07-24 | Dave McAlpin | Internationalization. Major revisions to 2.2 – 2.5. Harmonization of section 3. |
| wd-08 | 2003-09-30 | Gabe Wachob | New resolution section, addressed many issues |
| wd-09 | 2003-10-28 | All members of XRI Editors SC, Jerry Kindall | See sequence of 09 (a-m) drafts posted to XRI Editors SC. |
| wd-rc-1 | 2003-11-07 | Drummond Reed | Converted all final proofing action items from 09m to comments. |
| wd-rc-2 | 2003-11-19 | All Editors, Drummond Reed | Incorporated all comments and feedback, resolved all action items, updated all comments. |

# Appendix F. Notices

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

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