Extensible Resource Identifier (XRI) Version 3.0

Working Draft 03

18 January 2010

Specification URIs:
This Version:
http://docs.oasis-open.org/xri/xri/v3.0/WD01/xri-syntax-3.0-wd03.html
http://docs.oasis-open.org/xri/xri/v3.0/WD01/xri-syntax-3.0-wd03.doc
http://docs.oasis-open.org/xri/xri/v3.0/WD01/xri-syntax-3.0-wd03.pdf

Previous Version:
http://docs.oasis-open.org/[tc-short-name] / [additional path/filename] .html
http://docs.oasis-open.org/[tc-short-name] / [additional path/filename] .doc
http://docs.oasis-open.org/[tc-short-name] / [additional path/filename] .pdf

Latest Version:
http://docs.oasis-open.org/[tc-short-name] / [additional path/filename] .html
http://docs.oasis-open.org/[tc-short-name] / [additional path/filename] .doc
http://docs.oasis-open.org/[tc-short-name] / [additional path/filename] .pdf

Technical Committee:
OASIS Extensible Resource Identifier (XRI) TC

Chair(s):
Peter Davis, NeuStar
Drummond Reed, XDI.org

Editor(s):
Drummond Reed, XDI.org

Related work:
This specification replaces or supercedes:

- XRI Syntax 2.0 Committee Specification, April 2008
- XRI 1.0 Specification, January 2004

This specification is related to:

- XRI http: and https: Bindings 1.0
- XRI info: Binding 1.0
- XRD 1.0
- XRI Resolution 3.0

Declared XML Namespace(s):
None.

Abstract:
This document is the normative technical specification for XRI generic syntax and transformation, normalization, and comparison rules.

Status:

This document was last revised or approved by the XRI Technical Committee on the above date. The level of approval is also listed above. Check the “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

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

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

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

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

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

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

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

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' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on 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 implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS", [insert specific trademarked names and abbreviations here] are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Table of Contents

1 Introduction

XRI (Extensible Resource Identifier) provides a common language for structured identifiers that may be used to share semantics across protocols, domains, systems, and applications. XRI builds directly on the structure and capabilities of URI (Uniform Resource Identifier) **[URI]** and IRI (Internationalized Resource Identifier) **[IRI]**. XRI is a profile of URI and IRI syntax and normalization rules for producing URIs or IRIs that contain additional structure and semantics beyond those specified by **[URI]** or **[IRI]**.

1.1 Motivations

There are as many reasons for needing a common language for structured identifiers (XRI) as there are for needing a common language for structured data (XML). Some of the most commonly cited motivations are:

- To unambiguously assert that the same resource is being identified across different protocols, e.g., HTTP, HTTPS, FTP, SMTP, XMPP.
- To unambiguously identify the same resource in different contexts, i.e., within different domains, systems, applications, namespaces, etc.
- To assign, resolve, and determine the equivalence of different synonymous identifiers for the same resource, e.g., persistent vs. reassignable synonyms, human-readable vs. machine-friendly synonyms, localized vs. non-localized synonyms.
- To identify different versions of the same resource in a manner that is consistent across multiple domains, systems, and applications.
- To create structured identifiers to address, navigate, and share structured data, such as RDF graphs.

1.2 Related Work

1.2.1 URN (Uniform Resource Name)

URNs, identified by the URI scheme `urn:`, were standardized by the IETF in **[RFC2141]**. The motivation was to establish a class of resource identifiers that are persistent (assigned once to a resource and never reassigned to a different resource) and location-independent (do not change even if the resource moves to a different domain or location on a network). The relationship of URNs, URIs, and URLs is explained in **[RFC3305]**.

XRI architecture, like URI architecture, requires the capability to express both persistent and reassignable identifiers. Therefore, while an XRI is reassignable by default, XRI syntax includes the capability to express that an XRI, or a specific component of an XRI, is persistent. An XRI in which all components are expressed as persistent meets the functional requirements of a URN as specified in **[RFC1737]**.

1.2.2 Handle and DOI (Digital Object Identifier)

The Handle system is defined in **[RFC3650]**, **[RFC3651]**, and **[RFC3652]**. To quote from RFC 3650:

> "…the Handle System [is] a distributed information system designed to provide an efficient, extensible, and secured global name service for use on networks such as the Internet. The Handle System includes an open protocol, a namespace, and a reference implementation of the protocol. The protocol enables a distributed computer system to store names, or handles, of digital resources and resolve those handles into the information necessary to locate, access, and otherwise make use of the resources. These associated values can be changed as needed to reflect the current state of the identified resource without changing the handle. This allows the name of the item to persist over changes of location and other current state information."

RFC 3650 also provide the following description about how Handle relates to other identifier systems:

> "The Handle System crosses boundaries. Looked at as a name resolution system, it might be compared to DNS. If used to implement a URI/URN namespace, it could be used with any URI/URN scheme. If used for distributed information updates and administration, it could be considered a simplified-version of a distributed database system."

Although both Handle and XRI architecture involve global identification and resolution, the focus of XRI is to extend URI and IRI architecture to establish shared structured and semantics across multiple URI/IRI schemes, and to provide extensible resolution of XRIs using standard Web content types.

1.2.3 XRD (Extensible Resource Descriptor)

XRD (and its companion format, XRDS—Extensible Resource Descriptor Sequence) is a separate specification produced by the XRI Technical Committee. XRD and XRDS were first defined in the XRI Resolution 2.0 specification **[CoolURI]** D. Ayers, M. Völkel, *Cool URIs for the Semantic Web*, http://www.w3.org/TR/cooluris/, W3C Interest Group Note 03, December 2008.

[RFC3650] Sam Sun, Larry Lannom, Brian Boesch, *Handle System Overview*, http://hdl.handle.net/4263537/4069, RFC 3650, November 2003.

[RFC3651] Sam Sun, Sean Reilly, Larry Lannom, *Handle System Namespace and Service Definition*, http://hdl.handle.net/4263537/4068, RFC 3651, November 2003.

[RFC3652] Sam Sun, Sean Reilly, Larry Lannom, Jason Petrone, *Handle System Protocol (ver 2.1) Specification*, http://hdl.handle.net/4263537/4086, RFC 3652, November 2003.

[XRIRes2] as a simple, standard XML format for metadata describing a resource and its related resources.

To facilitate use of these formats with other protocols and applications besides XRI resolution, in late 2008 the XRI TC began work on XRD 1.0 as a separate standalone specification. [ref to XRD 1.0 CD 02] was published in [month 2010].

1.2.4 XRI Resolution

XRI resolution is the process of dereferencing an XRI to a descriptor of the resource it identifies (if such a descriptor is available). There is no requirement that a particular resolution protocol be used to resolve an XRI, nor that that it be dereferenceable. However the XRI TC publishes a standard resolution protocol based on the XRD and XRDS descriptor formats (above) that may be used for this purpose. XRI Resolution 3.0 [XRIRes3 ref to be supplied] is a companion to this specification.

1.2.5 XDI (XRI Data Interchange)

XDI is a separate OASIS Technical Committee at OASIS whose purpose is to define a structured data sharing format and protocol based on the RDF graph model using XRIs to address and describe the graph. The XDI 1.0 specifications are based on the XRI 3.0 specifications. See the XDI TC home page at http://www.oasis-open.org/committees/xdi.

1.3 Previous Versions

This section explains the relationship of this specification to the previous XRI specifications.

1.3.1 XRI 1.0

The first generation of the XRI specifications were published in January 2004. They defined both a syntax and a resolution protocol. Implementation experience with these specifications led to the XRI 2.0 activity begining in early 2005. Use of XRI 1.0 was deprecated once XRI 2.0 was completed.

1.3.2 XRI 2.0

The second generation of the XRI specifications consisted of XRI Syntax 2.0 Committee Specification 01 **[XRISyntax2]** in April 2008, and XRI Resolution 2.0 Committee Specification 01 **[CoolURI]** D. Ayers, M. Völkel, *Cool URIs for the Semantic Web*, http://www.w3.org/TR/cooluris/, W3C Interest Group Note 03, December 2008.

[RFC3650] Sam Sun, Larry Lannom, Brian Boesch, *Handle System Overview*, http://hdl.handle.net/4263537/4069, RFC 3650, November 2003.

[RFC3651] Sam Sun, Sean Reilly, Larry Lannom, *Handle System Namespace and Service Definition*, http://hdl.handle.net/4263537/4068, RFC 3651, November 2003.

[RFC3652] Sam Sun, Sean Reilly, Larry Lannom, Jason Petrone, *Handle System Protocol (ver 2.1) Specification*, http://hdl.handle.net/4263537/4086, RFC 3652, November 2003.

[XRIRes2] published in April 2008. These have been deployed in different services and code bases, including the XDI.org XRI Global Registry Service (http://www.xdi.org), the OpenXRI open source project (http://www.openxri.org), and OpenID 2.0 (http://www.openid.net).

1.4 Terminology

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]**.

1.5 Syntax Notation

This specification uses the syntax notation employed in **[IRI]**: 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 **[URI]**. As with URIs, the precise bit-and-byte representation of an XRI reference 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 section 3.1.

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

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

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

1.6 Normative References

[IRI] M. Dürst, M. Suignard, *Internationalized Resource Identifiers (IRIs)*, http://www.ietf.org/rfc/rfc3987.txt, RFC 3987, January 2005.

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

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

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

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

[RFC2718] L. Masinter, H. Alvestrand, D. Zigmond, R. Petke, *Guidelines for New URL Schemes*, http://www.ietf.org/rfc/rfc2718.txt, RFC 2718, November 1999.

[RFC2732] R. Hinden, B. Carpenter, L. Masinter, *Format for Literal IPv6 Addresses in URL's,* http://www.ietf.org/rfc/rfc2732.txt, RFC 2732, December, 1999.

[RFC3305] M. Mealing, R. Denenberg, *Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations*, http://www.ietf.org/rfc/rfc3305.txt, RFC 3305, August 2002.

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

[RFC3629] F. Yergeau, *UTF-8, A Transformation Format of ISO 10646,* http://www.faqs.org/rfcs/rfc3629.html, RFC 3629, November, 2003.

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

[URI] T. Berners-Lee, R. Fielding, L. Masinter, *Uniform Resource Identifier (URI): Generic Syntax,* http://www.ietf.org/rfc/rfc3986.txt, STD 66, RFC 3986, January 2005.

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

1.7 Non-Normative References

[CoolURI] D. Ayers, M. Völkel, *Cool URIs for the Semantic Web*, http://www.w3.org/TR/cooluris/, W3C Interest Group Note 03, December 2008.

[RFC3650] Sam Sun, Larry Lannom, Brian Boesch, *Handle System Overview*, http://hdl.handle.net/4263537/4069, RFC 3650, November 2003.

[RFC3651] Sam Sun, Sean Reilly, Larry Lannom, *Handle System Namespace and Service Definition*, http://hdl.handle.net/4263537/4068, RFC 3651, November 2003.

[RFC3652] Sam Sun, Sean Reilly, Larry Lannom, Jason Petrone, *Handle System Protocol (ver 2.1) Specification*, http://hdl.handle.net/4263537/4086, RFC 3652, November 2003.

[XRIRes2] G. Wachob, D. Reed, L. Chasen, W. Tan, S. Churchill, *Extensible Resource Identifier (XRI) Resolution V2.0*, http://docs.oasis-open.org/xri/2.0/specs/xri-resolution-V2.0.html, OASIS, April 2008.

[XRISyntax2] D. Reed, D. McAlpin, Extensible Resource Identifier (XRI) Syntax V2.0, http://www.oasis-open.org/committees/download.php/15376/xri-syntax-V2.0-cs.html, OASIS, April 2008.

2 XRI-to-URI/IRI Bindings

XRIs are referred to as *abstract identifiers* (see the *Glossary* in Appendix B) because they do not, by themselves, resolve directly to resource representations on the Internet or other digital networks. Rather an XRI is an identifier for an abstract resource (referred to as a “non-document resource” or “non-information resource” by the W3C Technical Architecture Group **[CoolURI]**) that must be bound to a specific *resolution context* before it can be resolved. This binding is achieved by appending the XRI to a URI or IRI to form another syntactically valid URI/IRI. The result is called a *bound XRI*.

Different types of bound XRIs are referred to by the scheme name of the URI/IRI to which the XRI is bound, e.g., http: XRI, https: XRI, xmpp: XRI, etc. The URI/IRI to which the XRI is bound is referred to as the *base URI/IRI*. The process of parsing the original XRI from the bound XRI is called *unbinding*, and when necessary to avoid ambiguity, the resulting XRI is referred to as an *unbound XRI*.

An XRI MAY be bound to any valid URI/IRI provided the binding produces a syntactically valid URI/IRI. However the resulting URI/IRI may not be recognizable as a bound XRI, either by humans or machines, unless that binding conforms to a formal specification.

Following are the requirements for an XRI binding specification:

- The specification SHOULD be named by the URI/IRI scheme name (or names, if the specification defines more than one binding), e.g., *XRI info: Binding Specification*.
- The specification SHOULD explain the motivations for this binding, and the applications for which this binding is recommended or preferred over other bindings.
- The specification MUST specify the ABNF rules for the binding.
- The specification MUST declare the normal form into which XRIs must be transformed prior to the binding. See section 4, *XRI Forms and Transformations*.
- The specification MUST define any other encoding rules that apply to XRIs prior to binding.
- The specification MAY include or reference a resolution specification that defines the resolution of XRIs bound according to the specification.
- The specification MUST include a conformance section [*ref: some language about conformance requirements – ref to SAML AuthN Context or Core where they talk about Profiles*].
- The specification MUST include a Security and Data Protection section.

XRI 3.0 includes the following binding specifications:

- *XRI http: and https: Binding 1.0.*
- *XRI mailto: Binding 1.0.*
- *XRI info: Binding 1.0.*

3 XRI Syntax

3.1 ABNF

The ABNF for XRI 3.0 builds on the ABNF for IRI as defined in **[IRI]**. The complete ABNF trees for both are included in this section for ease of reference.

3.1.1 XRI 3.0

Following is the normative ABNF syntax for XRI 3.0. Sections 3.2 through 3.4 explain this structure in more detail. Note that while the ABNF excerpts in those sections are copied from this section, if there are any descrepancies, the ABNF in this section is normative.

```
xri-reference     = xri
                  / relative-xri-ref

xri               = xri-hier-part [ "?" iquery ] [ "#" ifragment ]

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

relative-xri-part = xri-path-abs
                  / xri-path-noscheme
                  / ipath-empty

xri-hier-part     = xri-authority xri-path-abempty

xri-authority     = global-subseg *subseg

xri-path          = xri-path-abempty
                  / xri-path-abs
                  / xri-path-noscheme
                  / ipath-empty

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

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

xri-path-noscheme = xri-segment-nc *( "/" xri-segment )

xri-segment       = [ literal ] *subseg

xri-segment-nz    = ( literal / subseg ) *subseg

xri-segment-nc    = ( literal-nc / subseg ) *subseg

subseg            = global-subseg
                  / local-subseg
                  / xref

global-subseg     = gcs-char [ local-subseg / xref / literal ]

local-subseg      = lcs-char [ xref / literal ]

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

lcs-char          = "*" / "!"
```

```
xref              = "(" [ xref-value ] ")"

xref-value        = xri-reference
                  / iri

literal           = 1*xri-pchar

literal-nc        = 1*xri-pchar-nc

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

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

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

xri-gen-delims    = ":" / "/" / "?" / "#" / "[" / "]" / "(" / ")"
                  / gcs-char / lcs-char

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

3.1.2 IRI (RFC 3987)

The following ABNF from **[IRI]** is included here for reference only.

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

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

ihier-part        = "//" iauthority ipath-abempty
                  / ipath-abs
                  / ipath-rootless
                  / ipath-empty

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

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

ihost             = IP-literal / IPv4address / ireg-name

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

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

IPv6address       =                            6( h16 ":" ) ls32
                  /                       "::" 5( h16 ":" ) ls32
                  / [               h16 ] "::" 4( h16 ":" ) ls32
                  / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
                  / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
                  / [ *3( h16 ":" ) h16 ] "::"    h16 ":"   ls32
                  / [ *4( h16 ":" ) h16 ] "::"              ls32
                  / [ *5( h16 ":" ) h16 ] "::"              h16
                  / [ *6( h16 ":" ) h16 ] "::"

ls32              = ( h16 ":" h16 ) / IPv4address

h16               = 1*4HEXDIG

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

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

port              = *DIGIT

ipath-abempty     = *( "/" isegment )

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

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

ipath-empty       = 0<ipchar>

isegment          = *ipchar

isegment-nz       = 1*ipchar

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

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

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

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

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

pct-encoded       = "%" HEXDIG HEXDIG

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

reserved          = gen-delims / sub-delims

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

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

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

3.2 Hierarchical Structure

XRI follows the same generic hierarchical structure as URI/IRI. It supports the same query and fragment syntax, as well as the same distinction between absolute and relative references.

```
xri-reference     = xri
                  / relative-xri-ref

xri               = xri-hier-part [ "?" iquery ] [ "#" ifragment ]
```

```
relative-xri-ref  = relative-xri-part [ "?" iquery ] [ "#" ifragment ]

relative-xri-part = xri-path-abs
                  / xri-path-noscheme
                  / ipath-empty

xri-hier-part     = xri-authority xri-path-abempty
```

3.2.1 Authority

Like an absolute URI/IRI, an absolute XRI begins with an authority component. URI/IRI architecture defines a special syntax for this component. With XRI architecture, the authority segment uses the same subsegment structure as any other XRI segment with one exception: it MUST begin with a global subsegment (see section 3.3).

```
xri-authority     = global-subseg *subseg
```

3.2.2 Path

All segments in a relative XRI, and all segments after the authority segment in an absolute XRI, constitute the path component of the XRI. XRI path syntax options are the same as with URI/IRI syntax with the exception that if a relative XRI does not begin with a forward slash ("/"), the first path segment MUST NOT include a colon. This prevents a relative XRI from being interpreted as an absolute URI/IRI.

```
xri-path          = xri-path-abempty
                  / xri-path-abs
                  / xri-path-noscheme
                  / ipath-empty

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

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

xri-path-noscheme = xri-segment-nc *( "/" xri-segment )

ipath-empty       = 0<ipchar>
```

3.2.3 Query

The query component of XRI syntax is inherited directly from IRI syntax.

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

3.2.4 Fragment

The fragment component of XRI syntax is also inherited directly from IRI syntax.

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

3.3 Segment Structure

Besides scheme-independence, one of the fundamental differences between XRI and URI/IRI syntax is segment structure. URI/IRI architecture has no structure within segments; each segment is transparent to a URI/IRI processor. XRI segments are further structured into *subsegments*. Each subsegment is delimited by delimiter characters within a segment in the same way each segment is delimited by forward slashes within a path. Note that the first subsegment within a segment is the only subsegment that does not require a leading delimiter – it may be a literal (see section 3.3.5).

```
xri-segment       = [ literal ] *subseg
```

```
xri-segment-nz    = ( literal / subseg ) *subseg

xri-segment-nc    = ( literal-nc / subseg ) *subseg
```

3.3.1 Subsegments

There are three types of XRI subsegments: global, local, and cross-references (`xref` in the ABNF). Global subsegments must start with a global context symbol, and may be followed by either a local subsegment, a cross-reference, or a literal. Local subsegments must start with a local context symbol, and may be followed by either a cross-reference or a literal.

```
subseg            = global-subseg
                  / local-subseg
                  / xref

global-subseg     = gcs-char [ local-subseg / xref / literal ]

local-subseg      = lcs-char [ xref / literal ]
```

3.3.2 Global Context Symbols

There are four XRI global context symbols, chosen from among the valid subdelimiters in URI/IRI syntax.

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

The purpose of global context symbols is to define a small set of semantic contexts shared globally across all XRIs, as specified in Table 1.

Context Symbol	Context Name	Definition
=	Personal	Identifiers for which the ultimate authority is an individual person.
@	Organizational	Identifiers for which the ultimate authority is an organization.
+	General	Identifiers for which there is no ultimate authority because they represent generic dictionary concepts or “tags” whose meaning is determined by consensus. (In the English language, for example, these would be the generic nouns.)
$	Special	Identifiers for which the ultimate authority is a standards setting organization. Authority for identifiers assigned in this context is delegated by the OASIS XRI Technical Committee.

Table 1: Global Context Symbols

3.3.3 Local Context Symbols

There are two XRI local context symbols, again chosen from among the valid delimiters in URI/IRI syntax.

```
lcs-char          = "*" / "!"
```

The purpose of local context symbols is to define two universal characteristics of identifiers in any context, as specified in Table 2.

Context Symbol	Context Name	Definition
*	Reassignable/ Recyclable	The association between the identifier and the resource it identifies MAY change over time.
!	Persistent/ Non-recyclable	The association between the identifier and the resource it identifies MUST NOT change over time, i.e., the identifier meets the requirements of a URN as specified in **[RFC1737]**.

Table 2: Local Context Symbols

With XRI, identifier persistence is determined at the subsegment level – any particular subsegment may be persistent or reassignable. For an entire XRI to be persistent, all of its subsegments MUST be persistent, and it MUST NOT include a query or fragment component.

If a local subsegment begins with "!", it MUST be persistent. XRI authorities SHOULD use "!" as the delimiter for any local subsegment intended to be persistent.

A global subsegment is persistent if it consists of: a) a global context symbol alone, or b) a global context symbol followed by a persistent local subsegment.

If the XRI consists of multiple persistent subsegments assigned by different XRI authorities, each authority MUST independently meet the persistent requirements for the subsegment(s) for which it is authoritative.

3.3.4 Cross-References

Cross-reference syntax enables encapsulation of identifiers that would otherwise not be parsed as a single XRI subsegment (such as a URI or IRI) to be treated as a single XRI subsegment. Cross-references are one method XRI syntax provides to enable an identifier assigned in one context to be reused in another context, permitting sharing of identifiers across contexts. This simplifies identifying logically equivalent resources across different hierarchies (a directory concept referred to as "heterarchy" or "polyarchy").

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

```
xref              = "(" [ xref-value ] ")"

xref-value        = xri-reference
                  / iri
```

3.3.5 Literals

Literals are the atomic elements of XRI structure. They are the strings of non-delimiter path characters that ultimately identify a resource in a context.

```
literal           = 1*xri-pchar

literal-nc        = 1*xri-pchar-nc
```

3.4 Characters

The final ABNF rules governing XRI characters and encoding are are inherited from **[IRI]**, which is a superset of the generic URI syntax defined in **[URI]**. The primary difference between XRI and URI/IRI character rules is that the XRI global and local context symbol characters have been moved from the URI/IRI subdelimiter set into the XRI general delimiter set.

3.4.1 Path Characters

As with URI/IRI syntax, XRI path characters may be any character except general delimiters.

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

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

3.4.2 Reserved Characters

The reserved characters fall into general delimiters and subdelimiters.

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

If an XRI reserved character is used as a data character and not as a delimiter, the character MUST be percent-encoded per the rules in section 3.4.5. XRI references that differ in the percent-encoding of a reserved character MUST NOT be considered equivalent.

3.4.2.1 General Delimiters

XRI general delimiters are the URI/IRI general delimiter set plus the XRI global context symbols and local context symbols.

```
xri-gen-delims    = ":" / "/" / "?" / "#" / "[" / "]" / "(" / ")"
                  / gcs-char / lcs-char
```

3.4.2.2 Subdelimiters

XRI subdilimiters are the URI/IRI subdelimiters minus those allocated to the XRI general delimiter set.

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

3.4.3 Unreserved Characters

XRI has the same set of unreserved characters as **[IRI]**.

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

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
```

Percent-encoding unreserved characters in an XRI does not change what resource is identified by that XRI. However, it may change the result of an XRI comparison (see section [ref]), so unreserved characters SHOULD NOT be percent-encoded.

3.4.4 Character Encoding

The standard character encoding of XRI is UTF-8, as recommended by **[RFC2718]**. When an XRI reference is presented as a human-readable identifier (XRI native form, section 4.1.1), the representation of the XRI reference in the underlying document may use the character encoding of the underlying document (XRI native form, section 4.1.1). However, this representation must be converted to UTF-8 before the XRI can be processed outside the document (XRI normal form, section 4.1.2). This encoding into UTF-8 MUST include normalization according to Normalization Form KC (NFKC) as defined in **[UTR15]**. The stricter NFKC is specified rather than Normalization Form C (NFC) used in IRI encoding

[IRI] because NFKC reduces the number of UCS compatability characters allowed in an XRI and increases the probability of equivalence matches.

3.4.5 Percent-Encoded Characters

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

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

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

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

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

3.4.6 Excluded Characters

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

Note that presenting "space" or other whitespace characters in a non-percent-encoded form in a native XRI (see section 4.1.1) is not recommended 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 reference, so non-percent-encoded whitespace characters can make it difficult or impossible to determine where the identifier ends. Finally, non-percent-encoded whitespace can be used to maliciously construct subtly different identifiers intended to mislead the reader. For these reasons, non-percent-encoded whitespace characters SHOULD be avoided in presentation of native XRIs, and alternatives to whitespace as a logical separator character (such as a dot or hyphen) SHOULD be used whenever possible.

[IRI] provides the following additional guidance regarding other characters that should be avoided in IRIs. It applies equally to XRIs.

> "The UCS 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. This includes the full-width equivalents of Latin 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."

Finally, although they are not excluded characters, special care should be taken by user agents with regard to the display of UCS characters that are visual look-alikes (homographs) for XRI delimiters (all characters in the xri-reserved production, section **Error! Reference source not found.**). See section 7.2, "Spoofing and Homographic Attacks" for additional information.

4 XRI Forms and Transformations

To identify the same resource across multiple contexts, an XRI reference may need to appear in different forms. This section defines the valid forms of XRI references and the rules for transformations between them so consistent comparison of XRIs may be made across all contexts.

4.1 Forms

4.1.1 Native Form

An XRI reference is in *native form* (also called a *native XRI*) if is a string conforming to the ABNF in section 3.1.1 that uses any character set appropriate for display in a particular context. It is STRONGLY RECOMMENDED that display of native XRIs follow the guidance in section 3.4.6 regarding excluded characters. XRIs in native form SHOULD NOT be used for anything other than user display until they are transformed into one of the other forms below.

Because absolute XRIs begin with global context symbols, in some native contexts they may not be easily distinguished from strings that are not XRIs. In such cases, it is RECOMMENDED to place XRIs inside square brackets as a form of delimiter that is both human- and machine-readable.

```
[=example]
[@example]
[+example]
[$example]
```

4.1.2 XRI Normal Form

An XRI reference is in *XRI normal form* if it a native XRI that has been encoded according to the rules defined in section 4.2.1. This puts it in a standard form suitable for transport and comparison between contexts.

4.1.3 IRI Normal Form

An XRI reference is in *IRI normal form* if its XRI normal form has been encoded as a relative IRI and bound to a valid base IRI as defined in section 4.2.2. This form is suitable for any context that accepts IRIs.

4.1.4 URI Normal Form

An XRI reference is in *URI normal form* if its IRI normal form has been transformed into a valid URI reference according to the rules in section 0. This form is suitable for any context that accepts URIs.

4.2 Transformations

4.2.1 Native To/From XRI Normal Form

To transform an XRI reference in XRI native form into XRI normal form, the following steps MUST be performed in order:

1. Encode the native XRI into UTF-8 as defined in section 3.4.4.
2. Percent-encode the resulting string as defined in section 3.4.5.

To reverse this transformation:

1. Reverse the percent-encoding applied in step 2 above.
2. Encode the XRI reference in the desired character set for display.

4.2.2 XRI Normal Form To/From IRI Normal Form

To transform an XRI reference in XRI normal form into IRI normal form requires encoding the XRI reference as a relative IRI reference and then binding it to a base IRI. To do this, the following steps MUST be performed in order. In addition, because this transformation is not idempotent (i.e., it may yield a different result if applied more than once), this sequence of steps MUST be performed only once to avoid changing the semantics of the identifier

1. Percent-encode all IRIs contained within cross-references as required by the specification applicable for the relevant IRI scheme.
2. Percent-encode all percent “%” characters as “%25” across the entire XRI reference.
3. Percent-encode all number sign “#” characters that appear within a cross-reference (section 3.3.4) as “%23”.
4. Percent-encode all question mark “?” characters that appear within a cross-reference as “%3F”.
5. Percent-encode all slash “/” characters that appear within a cross-reference as “%2F”.
6. Bind the resulting relative IRI to a base IRI as defined by any valid XRI binding specification (section 2).

To reverse this transformation, reverse these steps. Again, this reversal is not idempotent, so it must be performed only once.

4.2.3 IRI Normal Form To/From URI Normal Form

An XRI reference in IRI normal form is transformed into URI normal form by following the algorithm for conversion of an IRI into a URI defined in section 3.1 of **[IRI]**.

To reverse this transformation, follow the steps defined in section 3.2 of **[IRI]**. Note that due to some inherent limitations of the percent-encoding specified in **[IRI]**, it is possible that an IRI converted to a URI and then reconverted into an IRI may differ from the original IRI. However in practice these limitations affect very few IRIs.

5 Relative XRI References

A relative XRI reference is an unbound XRI conforming to the `relative-xri-ref` ABNF rule in section 3.1.1.

Just as a relative URI/IRI reference is relative to a base URI/IRI which must be absolute, a relative XRI reference is relative to a base XRI which must be absolute. This base XRI may be in one of two forms: bound or unbound.

5.1 Resolution to a Bound Base XRI

A specified in section 2, a bound base XRI may be either an IRI or a URI. If the bound base XRI is an IRI, a relative XRI reference MUST be resolved into an absolute XRI reference as follows:

1. Encode the relative XRI reference into IRI normal form as defined in section 4.1.3.
2. Follow the relative reference resolution instructions in section 6.5 of **[IRI]**.

If the bound base XRI is a URI, a relative XRI reference MUST be resolved into an absolute XRI reference as follows:

1. Encode the relative XRI reference into URI normal form as defined in section 4.1.4.
2. Follow the relative reference resolution instructions in section 5 of **[URI]**.

IMPORTANT: The algorithms described in **[IRI]** and **[URI]** may produce incorrect results when applied to XRI references in XRI normal form, particularly when those XRI references contain cross-references. However these algorithms will apply to XRI references in XRI normal form provided that the processor:

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

5.2 Resolution to an Unbound Base XRI

If the base XRI is an unbound XRI, a relative XRI reference MUST be resolved into an absolute XRI reference as follows:

1. Encode the relative XRI reference into the same normal form as the base XRI (i.e., into either XRI normal form, IRI normal form, or URI normal form) as defined in section 4.1.
2. Append the relative XRI reference to the base XRI by direct concatenation.

IMPORTANT: Concatenation is specified because, unlike URIs and IRIs, XRIs contain structure within segments, and thus a relative XRI reference may be relative to another XRI subsegment and not just to an XRI segment.

6 Normalization and Comparison

In general, the normalization and comparison rules for generic IRIs and URIs specified in Section 5 of **[IRI]** and Section 6 of **[URI]** apply to XRIs. This section describes several 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.

Each application that uses XRI references 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, mapping reassignable XRIs to persistent XRIs).

6.1 Unbinding

Comparison of XRI references MUST be based on their unbound form. This means that prior to comparison the XRIs either must either be: a) unbound, or b) bound to equivalent base URIs or IRIs.

6.2 Case

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

- Comparison of authority components (section 3.2.1) and path components (section 3.2.2) is case-insensitive as defined in **[IRI]**. Note that this higher standard of case-insensitivity, applied to both the authority and path components, is due to the need for XRI comparisons to be interoperable across the widest possible range of contexts.
- As specified in section 3.4.5, comparison of characters in a percent-encoding construction is case-insensitive for the hexadecimal digits “A” through “F”, i.e. “%ab” is equivalent to “%AB”.

6.3 Encoding, Percent-Encoding, and Transformations

- Two XRI references 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 XRI references that differ only in whether unreserved characters are percent-encoded SHOULD be considered equivalent. If one XRI percent-encodes one or more unreserved characters, and another XRI differs only in that the same characters are not percent-encoded, they are equivalent.
- All forms of an XRI reference during the transformation processes described in section 4.2 SHOULD be considered equivalent.

6.4 Cross-References

- If an XRI reference contains a URI reference or IRI reference as a cross-reference, the URI reference or IRI reference SHOULD be reasonably canonical with respect to its scheme.
- If an XRI reference contains another XRI reference as a cross-reference, the rules in this section SHOULD be applied recursively to each cross-reference. For example, the following two XRIs should be considered equivalent:

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

6.5 Canonicalization

It is not possible for XRI references containing URI/IRI cross-references to have a canonical form since many URI/IRI schemes, including the http: scheme, do not define a canonical form. With this exception, an XRI MAY be transformed into a canonical form by following these steps:

1. Encode the XRI reference in XRI normal form as defined in section 4.1.2.
2. Remove percent encoding for all unreserved characters.
3. For all percent-encoded characters, normalize the hexadecimal digits “a” through “f” to UPPERCASE.
4. For all non-percent-encoded characters in the authority component (section 3.2.1) and path component (section 3.2.2), normalize to lowercase as defined in **[IRI]**.

Applications which deal exclusively or primarily with XRIs SHOULD specify use of this canonical form unless there is a compelling reason to do otherwise.

7 Security and Data Protection Considerations

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

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

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

In addition, note that section 2 requires that all XRI binding specifications include their own Security and Data Protection Considerations sections.

7.1 Cross-References

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

7.2 Spoofing and Homographic Attacks

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

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

To help prevent this problem, XRI registries SHOULD institute policies preventing the registration of deceptive XRIs. In particular, they should guard against spoofing the leading global context symbol character (section 3.3.2) with a homograph from the Unicode character set.

To help prevent this or any other attack based on spoofing legitimate XRI delimiters (all characters in the `xri-reserved` production, section 3.4.2), user agents SHOULD employ one or more of the following safeguards, particularly with regard to the authority segment of an XRI: a) visually distinguish the defined XRI delimiter characters using special color, size, font, or other mechanism that enables users to clearly understand when a legitimate XRI delimiter character is being displayed, b) do not display any homograph of any XRI delimiter character in unencoded form, and/or c) warn the user when an XRI contains a potentially deceptive homographic character.

7.3 UTF-8 Attacks

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

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

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

677 7.4 XRI Usage in Evolving Infrastructure

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

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

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

8 Conformance

An implementation is a *conforming XRI parser* if it meets the conditions in section 8.1. An implementation is a *conforming XRI library* if it meets the conditions in section 8.2.

8.1 XRI Parser

An implementation conforms to this specification as an XRI parser if it meets the following conditions:

1. It can validate an XRI reference according to the ABNF defined in section 3.1.1.
2. It can parse an XRI into any of the ABNF components defined in section 3.1.1.

8.2 XRI Library

An implementation conforms to this specification as an XRI parser if it meets the following conditions:

1. It includes a conformant XRI parser as defined in section 8.1.
2. It can perform XRI transformations as defined in section 4.2.
3. It can perform relative XRI resolution as defined in section 5.
4. It can perform XRI normalizations and comparisions as defined in section 6.

A. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

[Participant Name, Affiliation | Individual Member]

[Participant Name, Affiliation | Individual Member]

B. Revision History

Revision	Date	Editor	Changes Made
03	2010-01-18	Drummond Reed	Third draft: all sections content complete except the acknowledgements in Appendix A. Some front matter and internal references still to be added.
02	2009-08-20	Drummond Reed	Second draft: Section 3 (ABNF section) now complete. Sections 4-8 still TODO.
01	2009-05-20	Drummond Reed	First draft: includes all ABNF and partial text for the ABNF sections.