Extensible Resource Identifier (XRI) Resolution V2.0

Working Draft 09, 10 November 2005

Document identifier:
xri-resolution-V2.0-wd-09

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

Editors:
Gabe Wachob, Visa International <gwachob@visa.com>

Contributors:
Drummond Reed, Cordance <drummond.reed@cordance.net>
Dave McAlpin, Epok <dave.mcalpin@epok.net>
Chetan Sabnis, Epok <chetan.sabnis@epok.net>
Peter Davis, Neustar <peter.davis@neustar.biz>
Mike Lindelsee, Visa International <mlindels@visa.com>

Abstract:
This document defines both a standard and a trusted HTTP-based resolution mechanism for Extensible Resource Identifiers (XRIs), specifically XRIs conforming to *Extensible Resource Identifier (XRI) Syntax V2.0* **[XRISyntax]** or higher. For the set of XRIs defined to provide metadata about other XRIs, see the *Extensible Resource Identifier (XRI) Metadata V2.0* **[XRIMetadata]**. For a basic introduction to XRIs, see the non-normative *Introduction to XRIs* **[XRIIntro]**. For a detailed guide to XRI implementation, see the non-normative *XRI Implementor's Guide* **[XRIGuide]**.

Status:
This document was last revised or approved by the XRI Technical Committee on the above date. The level of approval is also listed above. Check the current location noted above for possible later revisions of this document. This document is updated periodically on no particular schedule.
Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/xri.
For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/xri/ipr.php.
The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/xri.

Table of Contents

1 Introduction

1.1 XRI Resolution Framework

Extensible Resource Identifiers (XRIs) provide a uniform syntax for abstract identifiers as defined in **[XRISyntax]**. Because XRIs may be used across a wide variety of communities and applications (as Web addresses, wireless addresses, database keys, filenames, directory keys, object IDs, XML IDs, tags, etc.), no single resolution mechanism may prove appropriate for all XRIs. However, in the interest of promoting interoperability, this specification defines a standard framework for XRI resolution consisting of five parts:

- *XRDs* (Extensible Resource Descriptors – section [2]) is a simple, flexible XML-based container for XRI resolution metadata or any other metadata about a resource.
- *Generic authority resolution* (section [3]) is a simple, flexible resolution protocol for the authority segment of an XRI that uses HTTP/HTTPS as a transport.
- *Trusted authority resolution* (section [4]) is an extension of the generic resolution protocol that uses SAML assertions to create a chain of trust between the participating authorities.
- *Local resolution* (section [5]) is an extension of generic resolution that allows authorities to publish descriptions of resources identified by the local part of an XRI.
- *Proxy resolution* (section [6]) provides backwards compatability with existing HTTP(S) infrastructure by specifying how HTTP(S) servers can act as XRI proxy resolvers.

Table [1] summarizes the key differences between these four types of XRI resolution services:

Service type	Resolves XRI authority segment	Resolves XRI path/query component	Returns signed responses	Redirects for non-XRI-aware clients
Generic authority resolution	Yes	No	No	No
Trusted authority resolution	Yes	No	Yes	No
Local resolution	No	Yes	No	Yes
Proxy resolution	Yes	Yes	Optional	Yes

Table 1: Comparison of the four services in the XRI resolution framework.

All of these protocols are extensible, as described in section [2.7]. In addition, other XRI resolution services or protocols may be defined by future versions of this specification or by other specifications. Versioning of this specification is defined in section [8.2].

1.2 XRI Implementor's Guide

To minimize the non-normative material in this specification, it does not include sequence diagrams or extensive examples of resolution requests and responses. This material is available in the non-normative *XRI Implementor's Guide* **[XRIGuide]**. The *Implementor's Guide* is highly recommended for Internet developers, architects, network and directory administrators, and anyone else who builds, installs, maintains, or administers XRI infrastructure.

1.3 Terminology and Notation

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “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.

```
Examples look like this.
```

`XML elements and attributes that appear in text look like this.`

Throughout this document, the following namespaces prefixes have the meanings defined in Table 2 whether or not they are explicitly declared in the example or text.

Prefix	XML Namespace	Reference
xs	http://www.w3.org/2001/XMLSchema	
saml	urn:oasis:names:tc:SAML:2.0:assertion	**[SAML]**
ds	http://www.w3.org/2000/09/xmldsig#	**[XMLDSig]**
xrds	xri://$xrds	This document
xrd	xri://$xrd*($v*2.0)	This document

Table 2: XML namespace prefixes used in this specification.

Terms used in this document are defined in the glossary in Appendix C of **[XRISyntax]**.

2 XRDs (Extensible Resource Descriptors)

To provide a straightforward, flexible resolution mechanism, the response to an XRI resolution request is a simple, extensible XML document called an Extensible Resource Descriptor (abbreviated “XRD”.) While this specification defines only the XRD elements necessary to support delegated resolution and redirection of XRIs, an XRD can easily be extended to publish any form of metadata about the resource it describes.

2.1 XRD Format

To provide explicit support for extensibility, XRDs are designed to be used inside a single-element container document (XRDS) that has its own independent XML namespace “xri://$xrds”. The XRD schema elements are in the XML namespace “xri://$xrd*($v*2.0)”. By maintaining separate namespaces, the XRDS namespace can remain constant while the XRD and other contained namespaces can be versioned. See section [2.8] for more about versioning of the XRD schema.

The formal XML Schema definitions of XRDS and XRD is provided in Appendix A. The following example instance document illustrates the elements and attributes defined in these schemas:

```
<XRDS xmlns=”xri://$xrds” xmlns:xrd=”xri://$xrd*($v*2.0)”>
    <xrd:XRD xrd:id=”1” version="$v*2.0">
        <xrd:Query>*foo</xrd:Query>
        <xrd:Expires>2005-05-30T09:30:10Z</xrd:Expires>
        <xrd:Synonym>xri://@!1</xrd:Synonym>
        <xrd:XSynonym>xri://!!4!5!6</xrd:Synonym>
        <xrd:ProviderID>
        urn:uuid:c9f812f3-6544-4e3c-874e-d3ae79f4ef7b
        </xrd:ProviderID>
        <xrd:Service>
            <xrd:Type>
            xri://$resolution*authority*generic*($v*2.0)
            </xrd:Type>
            <xrd:Type>
            xri://$resolution*authority*trusted*($v*2.0)
            </xrd:Type>
            <xrd:URI xrd:priority=”10”>
            http://xri.example.com
            </xrd:URI>
            <xrd:URI xrd:priority=”15”>
            http://xri2.example.com
            </xrd:URI>
            <xrd:URI>https://xri.example.com</xrd:URI>
            <xrd:ProviderID>
            urn:uuid:f0502a17-4503-4463-8516-f1225b330e4d
            </xrd:ProviderID>
        </xrd:Service>
        <xrd:Service>
            <xrd:Type>xri://$resolution*local*($v*2.0)</xrd:Type>
            <xrd:URI>http://xri.example.com</xrd:URI>
            <xrd:ProviderID>
            urn:uuid:f0502a17-4503-4463-8516-f1225b330e4d
            </xrd:ProviderID>
        </xrd:Service>
        <xrd:Service>
            <xrd:Type>xri://$resolution*uri</xrd:Type>
            <xrd:MediaType>image/jpeg</xrd:MediaType>
            <xrd:URI>http://pictures.xri.example.com</xrd:URI>
        </xrd:Service>
        <xrd:Service>
            <xrd:Type>http://example.com/some/service/v3.1</xrd:Type>
            <xrd:URI>http://example.com/some/service/endpoint</xrd:URI>
        </xrd:Service>
         Other XRD elements here
    </xrd:XRD>
    Other XRDS elements here
</XRDS>
```

Following are the elements and attributes in the XRD schema:

xrd:XRD

Has an “xrd:id” attribute to uniquely identify this element within the containing `xrd:XRDS` document. Also has a “version” attribute to identify the version of this specification to which this element and its children conform.

Comment [DSR1]: Change this based on DaveM's email about the XML ID spec

xrd:XRD/xrd:Query

0 or 1 per `xrd:XRD` element. Expresses the XRI or other identifier whose resolution results in this `xrd:XRD` element. For authority resolution in the XRI resolution framework, this must be a qualified subsegment of the authority component of the XRI.

xrd:XRD/xrd:Expires

0 or 1 per `xrd:XRD` element. The date/time, in the form of `xs:dateTime`, after which this XRD cannot be relied upon. To promote interoperability, this date/time value SHOULD use the UTC "Z" time zone and SHOULD NOT use fractional seconds. A resolver using this XRD MUST NOT use the XRD after the time stated here. A resolver MAY discard this XRD before the time indicated in this result. If the HTTP transport caching semantics specify an expiry time earlier than the time expressed in this attribute, then a resolver MUST NOT use this XRD after the expiry time declared in the HTTP headers per section 13.2 of **[RFC2616]**.

xrd:XRD/xrd:Synonym

0 or more per `xrd:XRD` element. Contains an equivalent XRI, IRI, or URI for the described resource assigned by the same authority producing the current XRD. Resolution of this synonym should produce the same XRD (with the exception of the `xrd:Query`, `xrd:Expires`, and `xrd:Synonym` elements—see section [2.5].) This element also has the global priority attribute (see section [2.4]).

xrd:XRD/xrd:XSynonym

0 or more per `xrd:XRD` element. Contains an equivalent XRI, IRI, or URI for the described resource assigned by a different authority than the authority producing the current XRD. Resolution of this synonym may produce a different XRD for the described resource (see section [2.5].) This element also has the global priority attribute (see section [2.4]).

xrd:XRD/xrd:ProviderID

0 or 1 per `xrd:XRD`. A unique identifier of type `xs:anyURI` for the authority that produced this XRD. The value of this element MUST be such that there is negligible probability that the same value will be assigned as an identifier to any other authority. Note that for authority resolution in the XRI resolution framework, the authority identified by this element is the *describing* authority (the producer of the current XRD), not the authority *described* by the XRD. The latter is specified in the `xrd:XRD/xrd:Service/xrd:ProviderID` element for a resolution service endpoint (see below).

xrd:XRD/saml:Assertion

0 or 1 per `xrd:XRD`. Optional for generic authority resolution (section [3]), but required for trusted authority resolution (section [4]). A SAML assertion from the *describing* authority (the one providing the XRD) that asserts that the information contained in the enclosing XRD is authoritative. Because the assertion is digitally signed and the digital signature encompasses the containing `xrd:XRD` element, it also provides a mechanism for the recipient to detect unauthorized changes since the time the XRD was published.

Note that while a `saml:Issuer` element is required within a `saml:Assertion` element, this specification makes no requirement as to the value of the `saml:Issuer` element. It is up to the XRI community resolution root to place restrictions, if any, on the `saml:Issuer` element. A suitable approach is to use an XRI in URI-normal form that describes the community root authority. See section [2.3].

xrd:XRD/xrd:Service

0 or more. Describes a capability of the described resource, e.g., a network endpoint for performing further resolution, obtaining further metadata, or interacting directly with the described resource. This element also has the global priority attribute (see section [2.4]).

xrd:XRD/xrd:Service/xrd:Type

0 or more per `xrd:Service` element. A unique identifier of type `xs:anyURI` that identifies the type of capability being described. See section [2.2] for the resolution service types defined in this specification.

xrd:XRD/xrd:Service/xrd:MediaType

0 or more per `xrd:Service` element. The media type of content available at this service endpoint. The value of this element must be of the form of a media type defined in **[RFC2046]**. This element may appear multiple times to indicate all the media types available through this service endpoint. However the service endpoint MAY support additional media types not specified by instances of this element. If this element is not present, then a processor of the XRD SHOULD NOT make any assumption about the type of resource available at this endpoint.

xrd:XRD/xrd:Service/xrd:Pattern

0 or more per `xrd:Service` element. Of type `xs:string`. Contains a regular expression that XRD authorities may provide so an XRI resolver or other XRD processor can select a Service element on the basis of the local part of the XRI. See section [5.9].

xrd:XRD/xrd:Service/xrd:URI

0 or more per `xrd:Service` element. Of type `xs:anyURI`. If present, it indicates a transport-level URI for access to the capabilitiy described by the parent Service element. For the XRI resolution service types defined in section [2.2], this URI MUST be an HTTP or HTTPS URI. Other services may use other transport protocols. This element has the global priority attribute (see section [2.4].

xrd:XRD/xrd:Service/xrd:ProviderID

0 or 1 per `xrd:Service` element. Optional for generic authority resolution (section [3]), but required for trusted authority resolution (section [4]). See the definition for `xrd:XRD/xrd:ProviderID` above. In trusted resolution, when a resolution request is made to the authority at this service endpoint, the contents of the `xrd:XRD/xrd:ProviderID` element in the response MUST match the content of this element for correlation. See section [4.6]. The same usage MAY apply to other services not defined in this specification.

xrd:XRD/xrd:Service/ds:KeyInfo

0 or 1 per `xrd:Service` element. Optional for generic authority resolution (section [3]), but required for trusted authority resolution (section [4]). Provides the digital signature metadata needed to validate an XRD provided as a resolution response by the described authority. This element comprises the key distribution method for trusted authority resolution in the XRI resolution framework—see section [4.6].

XRD 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 XRI resolution schemes, new service description schemes, or other metadata about the described resource. See section [2.7].

2.2 XRI Resolution Metadata for XRDs

As defined in section 2.2.1.2 of **[XRISyntax]**, the GCS symbol “$” is reserved for special identifiers assigned by XRI TC specifications, other OASIS specifications, or other standards bodies. (See also **[XRIMetadata]**.) Within the “$” namespace, the XRI “$resolution” is reserved for identifiers assigned by this specification for the purposes of XRI resolution. Table 3 summarizes these XRIs.

Comment [DSR2]: Question: "$resolution" and the strings below could be considered long for equivalence comparisions, which will be done constantly in XRD processing. Should we consider going back to abbreviations like "$res" as used in CD01?

XRI	Use	See Section
xri://$resolution*authority	XRI namespace for authority resolution protocol types	[3]

xri://$resolution*authority*generic	XRI namespace for generic authority resolution	[3]
xri://$resolution*authority*generic*($v*2.0)	Version 2.0 of above	[3]
xri://$resolution*authority*trusted	XRI namespace for trusted authority resolution	[4]
xri://$resolution*authority*trusted*($v*2.0)	Version 2.0 of above	[4]
xri://$resolution*local	XRI namespace for local resolution	[5]
xri://$resolution*local*generic	XRI namespace for generic local resolution	[5]
xri://$resolution*local*generic*($v*2.0)	Version 2.0 of above	[5]
xri://$resolution*proxy	XRI namespace for HTTP(S) proxy resolution	[6]
xri://$resolution*proxy*($v*2.0)	Version 2.0 of above	[6]
xri://$resolution*uri	Special processing instruction to a proxy resolver to return an URI	[6]
xri://$resolution*xrd	Special processing instruction to a proxy resolver to return an XRD	[6]

Table 3: Special XRIs reserved for XRI resolution.

Using the standard extensibility mechanisms of XRI described in **[XRISyntax]**, the "$resolution" namespace may extended by other authorities besides the XRI Technical Committee. See **[XRIMetadata]** for more information about extending "$" namespaces.

2.3 XRI Encoding Requirements for XRDs

The W3C XML specifications [need ref] requires values of XML elements of type "anyURI" to be valid IRIs. However a further restriction applies to XRIs used in XRDs: because XRI resolution relies on HTTP or HTTPS as transport protocols, when an XRI is used as the value of an element in an XRD of type "anyURI", it MUST be in URI-normal form as defined in section 2.3 of **[XRISyntax]**.

Note that the XRIs in Table 3 are composed entirely of valid URI characters that do not require escaping in the transformation to URI-normal form. However XRIs that use characters only valid in IRIs or that use certain XRI syntax delimiters may require percent encoding in the transformation to URI-normal form as explained in section 2.3 of **[XRISyntax]**.

2.4 XRD Attribute Usage

2.4.1 ID Attributes

[Open issue: Need text here governing use of ID attributes. Should we mandate use of the XML ID specification? We also need to specify/clarify how an ordered list of XRDs should be assembled by lookahead or proxy resolvers. See Resolution Issue #16: ID Attribute at http://wiki.oasis-open.org/xri/Xri2Cd02/ReSolution/I16IdAttribute].

Comment [DSR3]: See text.

2.4.2 Version Attributes

[Open issue: Need text here governing use of version attributes. See Resolution Issue #14: Version Attribute at http://wiki.oasis-open.org/xri/Xri2Cd02/ReSolution/I14VersionAttribute. Note that this section should either reference or incorporate part of section 8.2].

Comment [DSR4]: See text.

2.4.3 Priority Attributes

During the XRI resolution process, four elements (Synonym, XSynonym, Service, and URI) may need to be selected from an XRD, yet may be present multiple times. The global "priority" attribute accepts a non-negative integer value that allows XRI authorities to provide guidance to XRI clients in selection of an element instance. Patterned on the priority attribute of records in DNS, this design simplifies offering redundant services.

The processing rules for the priority attribute are common across all services in the XRI resolution framework. They apply whenever there is more than one instance of the desired element type in an XRD (if there is only one instance, the priority attribute is ignored). The rules are:

1. The client SHOULD select the element instance with the lowest numeric value of the priority attribute. For example, an element with priority attribute value of "10" should be selected before an element with a priority attribute value of "11", and an element with priority attribute value of "11" should be selected before an element with a priority attribute value of "15". (Zero is the highest priority attribute value.)
2. If two or more instances of the same element type have identical priority attribute values, or if the priority attribute is not present, the client SHOULD select one of the instances at random.

An element selected according to these rules is referred to as "the highest priority element". If this element is eliminated from the set of matching elements, the next element selected according to these rules is referred to as "the next highest priority element". If a protocol operation specifying selection of the highest priority element fails, the client SHOULD attempt to select the next highest priority element unless otherwise specified. This process SHOULD be continued for all other element instances until success is achieved or all instances are exhausted.

When setting priority attributes, it is recommended that XRI authorities follow the practice common in DNS and set the default highest priority attribute value to "10".

2.5 XRI Synonym Usage

The XRI resolution framework includes the concept of *synonyms*. A synonym is an XRI that is not character-for-character equivalent with another XRI, but which identifies the same target resource (either the same concrete network resource, or an equivalent logical copy of this resource.) For example, when queried for a reassignable XRI, an authority may provide a persistent XRI as a synonym for future lookups.

In an XRI resolution response, an authority may provide synonyms for the query XRI using either the `xrd:XRD/xrd:Synonym` or `xrd:XRD/xrd:XSynonym` elements (section [2.1].) The value of either of these elements MAY be either an absolute XRI or a relative XRI reference as defined in section [need ref] of **[XRISyntax]**. If relative, the synonym is relative to any valid base XRI for the describing authority (the authority identified by the `xrd:XRD/xrd:ProviderID` element of the XRD—see section [2.1].)

The following sections explain the differences between these two elements.

2.5.1 Authoritative Synonyms (xrd:Synonym)

An authoritative synonym is an XRI for the described resource assigned directly by the describing authority (i.e., the authority identified by the `xrd:XRD/xrd:ProviderID` element of the XRD.) A common example is a persistent XRI assigned to a resource that has one or more reassignable XRIs.

If the describing authority is authoritative for a synonym, it MUST be expressed using the `xrd:XRD/xrd:Synonym` element. Resolution of an authoritative synonym SHOULD return the same XRD as the current query XRI except for the value of the `xrd:XRD/xrd:Query`, `xrd:XRD/xrd:Expires`, and `xrd:XRD/xrd:Synonym` elements.

2.5.2 Cross-Reference Synonyms (xrd:XSynonym)

A cross-reference synonym is an XRI for the described resource that is NOT assigned by the describing authority, but by some other authority. By definition, such an XRI serves as a cross-reference—see section 2.2.2 of **[XRISyntax]**.

If a synonym is a cross-reference, it MUST be expressed using the `xrd:XRD/xrd:XSynonym` element. Resolution of a cross-reference synonym MAY return a different XRD for the described resource, i.e., an XRD containing other synonyms, service endpoints, or other metadata describing the target resource. (Note however that such an XRD MAY be an XRI redirect back to the current query XRI (see section [3.2.7]), in which case it will produce the current XRD.)

2.6 Community Root XRDs

Identifier management policies are defined on a community-by-community basis. With XRIs, the resolution community is specified by the first (leftmost) subsegment of the authority segment. This is referred to as the *community root authority*. When a resolution community chooses to create a new community root authority, it SHOULD define policies 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.

With XRIs, a community root authority may be an XRI authority or an IRI authority as specified in section 2.2.1 of **[XRISyntax]**. For an XRI authority, the community root may be either a global context symbol (GCS) character or top-level cross-reference as specified in section 2.2.1.1 of **[XRISyntax]**. In either case, the corresponding root XRD (or its equivalent) specifies the top-level authority resolution service endpoints for that community.

This community root XRD, or its location, is known *a priori* and is part of the configuration of a resolver, similar to the specification of root DNS servers in a DNS resolver. Note that is not strictly necessary to publish this information in an XRD—it may be supplied in any format that enables configuration of the XRI resolvers in the community. However, providing an XRD at a known location simplifies this process. In addition, for trusted resolution (section [4]), a recommended best practice is for the community root authority to publish an XRD containing a valid self-signed SAML assertion. (See also the recommendation about SAML assertion elements in section [2.1].)

If the first subsegment of an XRI authority is a GCS character and the following subsegment does not begin with a "*" (indicating a reassignable subsegment) or a "!" (indicating a persistent subsegment), then a "*" is implied and must be added when constructing the qualified subsegment as specified in section [3.2.3]. Table 4 and Table 5 illustrate the differences between parsing a reassignable subsegment following a GCS character and parsing a cross-reference, respectively.

XRI	xri://@example*internal/foo
XRI Authority	@example*internal
Community Root Authority	@
First Qualified Subsegment Resolved	*example

Table 4: Parsing the first subsegment of an XRI that begins with a global context symbol.

XRI	xri://(http://www.example.com)*internal/foo

XRI Authority	(http://www.example.com)*internal
Community Root Authority	(http://www.example.com)
First Qualified Subsegment Resolved	*internal

Table 5: Parsing the first subsegment of an XRI that begins with a cross-reference.

2.7 Extensibility

2.7.1 Extensibility of XRDs

Both the XRDS and the XRD schema in Appendix [A] use an an open-content model that is designed to be extended with other metadata. In most places, extension elements and attributes from namespaces other than "xri://$xrd*($v*2.0)" are explicitly allowed. These extension points are designed to simplify default processing of XRDs using a "Must Ignore" rule. The base rule is that unrecognized elements and attributes, and the content and child elements of unrecognized elements, MUST be ignored. As a consequence, elements that would normally be recognized by a processor MUST be ignored if they appear as descendants of an unrecognized element.

Extension elements MUST NOT require new interpretation of elements defined in this document. If an extension element is present, a processor MUST be able to ignore it and still correctly process the XRD document.

Extension specifications MAY simulate "Must Understand" behavior by applying an "enclosure" pattern. Elements defined by the XRD schema in Appendix [A] whose meaning or interpretation is modified by extension elements can be wrapped in a extension container element defined by the extension specification. This extension container element SHOULD be in the same namespace as the other extension elements defined by the extension specification.

Using this design, all elements whose interpretations are modified by the extension will now be contained in the extension container element that will be ignored by clients or other applications unable to process the extension. The following example illustrates this pattern using an extension container element from an extension namespace ("`other:SuperService`") that contains an extension element ("`other:ExtensionElement`"):

```
<XRD>
  <other:SuperService>
    <Service>
      …
      <other:ExtensionElement>…</other:ExtensionElement>
    </Service>
  </other:SuperService>
  <Service>
    …
  </Service>
</XRDS>
```

In this example, the `other:ExtensionElement` modifies the interpretation or processing rules for the parent `xrd:Service` element and therefore must be understood by the consumer for the proper interpretation of the parent `xrd:Service` element. To preserve the correct interpretation of the `xrd:Service` element in this context, the `xrd:Service` element is "wrapped" so only consumers that understand elements in the `other:SuperService` namespace will attempt to process the `xrd:ProviderID` element.

2.7.2 Other Points of Extensibility

The use of HTTP and XML in the design of XRDs and the XRI resolution framework provides additional specific points of extensibility:

- Specification of new resolution service types or other service types using XRIs, IRIs, or URIs as values of the `xrd:XRD/xrd:Service/xrd:Type` element.
- HTTP negotiation of content types, language, encoding, etc.
- Use of HTTP redirects (3XX) or other response codes defined by **[RFC2616]**.
- Use of cross-references within XRIs, particularly for associating new types of metadata with a resource.

2.8 Versioning

Versioning of the XRI specification set is expected to occur infrequently. Should it be necessary, this section describes versioning guidelines.

2.8.1 Version Numbering

Specifications from the OASIS XRI Technical Committee use a Major and Minor version number expressed in the form *Major.Minor*. The version number $Major_B.Minor_B$ is higher than the version number $Major_A.Minor_A$ if and only if:

$$Major_B > Major_A \text{ OR } ((Major_B = Major_A) \text{ AND } Minor_B > Minor_A)$$

Comment [DSR5]: Editor's reminder: we may want to use this definition to update the version comparision text in XRI Metadata.

2.8.2 Versioning of the XRI Resolution Specification

New releases of the XRI Resolution specification may specify changes to the resolution protocols and/or XRDs. When changes affect either of these, the resolution service type version number will be changed. Where changes are purely editorial, the version number will not be changed.

In general, if a change is backward-compatible, the new version will be identified using the current major version number and a new minor version number. If the change is not backward-compatible, the new version will be identified with a new major version number.

2.8.3 Versioning of XRDs

The `xrd:XRDS` document element is intended to be a completely generic container, i.e., to have no specific knowledge of the elements it may contain. Therefore it has no version element, and can remain stable indefinitely as there is no need to version its namespace.

The `xrd:XRD` element has an optional version attribute. When used, the value of this attribute MUST be the version value of the XRI Resolution specification to which its containing elements conform.

When new versions of the XRI Resolution specification are released, the namespace for the XRD schema may or may not be changed. If there is a major version number change, the namespace for the `xrd:XRD` schema is likely to change. If there is only a minor version number change, the namespace for the `xrd:XRD` schema may remain unchanged.

In general, maintaining namespace stability and adding to or changing the content of a schema are competing goals. While certain design strategies can facilitate such changes, it is difficult to predict how existing implementations will react to any given change, making forward compatibility difficult to achieve. Nevertheless, the right to make such changes in minor revisions is reserved. Except in special circumstances (for example, to correct major deficiencies or to fix errors), implementations should expect forward-compatible schema changes in minor revisions, allowing new messages to validate against older schemas.

Implementations SHOULD expect, and be prepared to deal with, new extensions and message types in accordance with the processing rules laid out for those types. Minor revisions may introduce new types that leverage the extension facilities described in Section [2.7]. Older implementations SHOULD reject such extensions gracefully when they are encountered in contexts with specific semantic requirements.

2.8.4 Versioning of Protocols

The protocols defined in this document may also be versioned by future releases of the XRI Resolution specification. If these protocols are not backward-compatible with older implementations, they will be assigned a new XRI for use in identifying their service type in XRDs. See section [2.2].

Note that it is possible for version negotiation to happen in the protocol itself. For example, HTTP provides a mechanism to negotiate the version of the HTTP protocol being used. If and when an XRI resolution protocol provides its own version-negotiation mechanism, the specification is likely to continue to use the same XRI to identify the protocol as was used in previous versions of the XRI Resolution specification.

3 Generic Authority Resolution

3.1 Introduction

In the XRI resolution framework, generic authority resolution is the process of looking up XRDs for each authority identified in the authority segment of an XRI in order to ultimately obtain the XRD for the final target authority. This process is directly analogous to the process of resolving each of the domain names in a federated DNS name in order to reach the target host. Besides the use of XRDs, the other key difference between DNS and XRI resolution is that the latter uses HTTP/HTTPS as the transport protocol. Note that future versions of this specification or specifications for other forms of XRI resolution may use other protocols.

3.1.1 Assumptions

The generic resolution protocol makes the following minimal assumptions about the XRIs being resolved:

- The endpoints representing the top-level authority for any absolute XRI are identified by the authority segment ("xri-authority" or "i-authority" productions) of the XRI as defined in section 2.2.1 of [XRISyntax].
- Only absolute XRIs can be resolved using this authority resolution protocol. To resolve a relative XRI reference, it must be converted into an absolute XRI using the procedure defined in section 2.4 of [XRISyntax].
- The XRI being resolved has been converted into URI-normal form, following the rules in section 2.3.1 of [XRISyntax].
- A resource represented by a single XRI may be accessed by multiple protocols at multiple service endpoints. For example, it is possible that a resource represented by a single XRI may be accessed through multiple HTTP URIs, or through both HTTP and another network protocol. While only HTTP or HTTPS access to resources is defined by this specification, XRDs enable access to resources via URIs in any scheme.
- Each network endpoint associated with a resource identified by an XRI may present a different subset, type, or representation of data or metadata associated with the identified resource. For example, two separate HTTP URIs may be associated with a single XRI, one for data access and the other for metadata access. As described in section [2.1] XRDs allow XRI authorities to define multiple service types using extensible descriptor fields based on service type, content type, XRI path, and URI semantics.

3.1.2 XRI vs. IRI Authorities

As described in section 2.2.1 of **[XRISyntax]**, XRI authorities and IRI authorities have different syntactic structures, partially due to the higher level of abstraction represented by XRI authorities. For this reason, XRI authorities are resolved to XRDs one subsegment at a time, as described in section [3.2]. IRI authorities, since they are based on DNS names or IP addresses, are resolved into an XRD through a special HTTP(S) request based on the DNS name or IP address identified by the IRI authority segment as described in section [3.3].

3.2 XRI Authority Resolution

3.2.1 Service Type

The following sections define the resolution service with the `xrd:XRD/xrd:Service/xrd:Type` element value “xri://$resolution*authority*generic*($v*2.0)”. See section [2.2].

3.2.2 Protocol

Following are the normative requirements for client and server behavior in generic authority resolution:

3. Resolution of each authority subsegment after the community root subsegment MUST proceed in subsegment order (left-to-right) using fully qualified subsegment values as defined in section [3.2.3].
4. A client MAY request lookahead resolution of multiple subsegments as defined in section [3.2.4].
5. Each subsegment MUST be resolved via an HTTP or HTTPS GET request to a URI constructed as defined in section [3.2.5].
6. Subsegments using XRI cross-reference syntax MUST be resolved as defined in section [3.2.6].
7. The HTTP(S) request MUST contain an Accept header with the value of “`application/xrd+xml`”.
8. The ultimate HTTP(S) response for a successful resolution request MUST contain either: a) a 2XX response with a valid XRD document (section [2.1]), or b) a 304 response signifying that the cached version on the client is still valid (depending on the client’s HTTP(S) request). 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]**.
9. Any ultimate response besides an HTTP 2XX or 304 SHOULD be considered an error in the resolution process. In lookahead or proxy resolution, such an error MUST be returned to the client as specified in section [7.1].
10. A successful response that constitutes an XRI redirect SHOULD be processed as defined in section [3.2.7].
11. All other use of HTTP(S) in this protocol MUST comply with the requirements in section [7]. In particular, HTTP caching semantics SHOULD be leveraged to the greatest extent possible to maintain the efficiency and scalability of the HTTP-based resolution system. The recommended use of HTTP caching headers is described in more detail in section [7.2.1].

3.2.3 Qualified Subsegments

A qualified subsegment is a subsegment as defined by the productions whose names start with “xri-subseg” in section 2.2.3 of **[XRISyntax]** *including the leading syntactic delimiter* (“*” or “!”). A qualified subsegment MUST include the leading syntatic delimiter even if it was optionally omitted in the original XRI (see section 2.2.3 of **[XRISyntax]**).

The first (or leftmost) subsegment of the XRI authority segment specifies the root of the identifier community. This subsegment is discussed in section [2.6].

The qualified subsegment immediately to the right of the community root is resolved in the context of the root, and all subsequent subsegments are resolved in the the context of the subsegment immediately to their left. XRI authority resolution is complete when an XRI resolver has completed the chain of XRDs for all subsegments in the XRI authority segment.

3.2.4 Lookahead Resolution

Both generic and trusted authority resolution services allow a client to request resolution of multiple authority subsegments in one transaction. This process is called *lookahead resolution*. If a client makes such a request, the responding resolution service MAY perform the additional lookahead resolution steps requested. In this case the responding service acts as a client to the other authority resolution service endpoints that need to be queried for the lookahead segments. Alternatively, the responding service may retrieve XRDs only from its local cache until it reaches a subsegment whose XRD is not locally cached, or it may simply lookahead only as far as it is authoritative. Any of these behaviors are reasonable, as are others not described here.

If an authority resolution service performs any lookahead resolution, it MUST return an ordered list of `xrd:XRD` elements in an `xrd:XRDS` document. Each XRID MUST correspond to a subsegment resolved by the service on behalf of the resolving client. The list of `xrd:XRD` elements in the `xrd:XRDS` document MUST appear in the same order as the subsegments in the original request. The responding service MAY resolve fewer subsegments than requested by the client. The responding service is under no obligation to resolve more than the first subsegment (for which it is, by definition, authoritative).

Comment [DSR6]: Need closure on XRD ordering issue using id attribute.

If the responding service does not resolve the entire set of subsegments requested, the resolving client MUST continue the authority resolution process itself. At any stage, however, the resolving client MAY request that the next authority resolution service resolve any additional unresolved subsegments.

3.2.5 Construction of the Next Resolution URI

At each step in authority resolution, a URI must be constructed for the next HTTP(S) request. This URI is constructed by the XRI resolver from two strings—one selected from the current XRD, and one representing the next subsegment or group of subsegments in the authority segment being resolved.

12. Select the highest priority `xrd:XRD/xrd:Service` element whose child `xrd:XRD/xrd:Service/xrd:Type` element value matches the following string:

    ```
    xri://$resolution*authority*generic*($v*2.0)
    ```

13. If no match is found, repeat the previous step for the following strings in the order they are listed until a match is found. If no match is found, return [*insert ref to correct error message for this condition*]:

    ```
    xri://$resolution*authority*generic
    xri://$resolution*authority
    xri://$resolution
    ```

14. For the selected Service element, select the highest priority URI element that contains an HTTP or HTTPS URI (depending on the desired transport protocol.)

Comment [DSR7]: Question: do we need this step? It is intended to help simplify the process of supporting multiple versions, and providing simple resolution services, but it may not be needed.

If the service endpoint URI does not end with a forward slash (“/”), one MUST be appended before proceeding.

The second string is called the *query XRI*. In authority resolution (both generic and trusted), the query XRI is supplied by the XRI client and consists of either:

- The next fully qualified subsegment to be resolved (see section [3.2.1]), or
- In the case of lookahead resolution, the next fully qualified subsegment plus the additional subsegments for which lookahead resolution is requested (see section [3.2.2]).

The query XRI MUST be in URI-normal form as required in section [2.2].

The final step is to append the query XRI to the path component of the service endpoint URI. The resulting URI is called the *next resolution URI*.

Construction of the next resolution URI is more formally described in this pseudo-code for resolving a “subsegment-list” via an HTTP URI called “sep-uri”:

```
res-uri = authority-res-uri

if (path portion of res-uri does not end in “/”):
    append “/” to path portion of res-uri

if (subsegment-list is not preceded with “*” or “!” separator):
    prepend “*” to subsegment-list

append uri-escape(subsegment-list) to path portion of res-uri
```

3.2.6 Cross-References

Any subsegment within an XRI authority segment may be a cross-reference (see section 2.2.2 of **[XRISyntax]**.) Cross-references are resolved identically to any other subsegment because the cross-reference is considered opaque by generic XRI resolution. In other words, the value of the cross-reference (including the parentheses) is the literal value of the subsegment for the purpose of authority resolution.

The one exception is a cross-reference rooted on the GCS dollar sign ("$"). The significance of such a cross-reference for resolution depends on the specification that defines the value of the identifier following the $ character. For the XRI suite of specifications, the significance of $ cross-references is defined by the *XRI Metadata Specification* **[XRIMetadata]**.

For example, an annotation cross-reference that begins with the GCS dollar sign ("$") followed by the hyphen character ("-"), is specified in **[XRIMetadata]** as insignificant, so this cross-reference and the delimiter that precedes it MUST be ignored entirely during resolution. A cross-reference that begins with the GCS dollar sign ("$") followed by the letter “v”, on the other hand, is version metadata that is defined in **[XRIMetadata]** as significant for resolution, so this should be treated as a standard cross-reference.

Table 6 provides several examples of resolving cross-references. In each example, subsegment “!b” resolves to a next resolution URI of “http://example.com/xri-authority/”, and lookahead resolution is not being requested.

Cross-reference type	Example XRI	Next Resolution URI after resolving “xri://@!a!b”
Absolute XRI	xri://@!a!b!(@!1!2!3)*e/f	http://example.com/xri-authority/!(@!1!2!3)
Absolute URI	xri://@!a!b*(mailto:jd@example.com)*e/f	http://example.com/xri-authority/*(mailto:jd@example.com)
Relative XRI	xri://@!a!b*(c*d)*e/f	http://example.com/xri-authority/*(c*d)
Relative URI	xri://@!a!b*(foo/bar)*e/f	http://example.com/xri-authority/*(foo%2fbar)
Metadata XRI (significant)	xri://@!a!b*($v/2.0)*e/f	http://example.com/xri-authority/*($v*2.0)
Metadata XRI (ignored)	xri://@!a!b*($-comment)*e/f	http://example.com/xri-authority/*e

Table 6: Examples of the Next Resolution URIs constructed using different types of cross-references.

3.2.7 XRI Redirects

An XRI authority resolution request may return an XRD containing no `xrd:XRD/xrd:Service` elements that advertise further resolution service endpoints but one or more `xrd:XRD/xrd:XSynonym` elements. This is called an "XRI redirect" because the XRD is effectively redirecting to a different XRI authority through the use of a cross-reference synonym (section [2.5.2]).

In this case, the unresolved portion of the original XRI being resolved (i.e., the remaining authority subsegments and any path and/or query component), is appended to the contents of the highest priority `xrd:XRD/xrd:XSynonym` element (called the *redirect synonym*). This resulting XRI is then resolved in place of the original XRI.

The one exception is if the original XRI contains additional unresolved authority subsegments and the redirect synonym contains a local path component. In this case the client SHOULD consider this an error condition and fail. An example is the following resolution response for the "*home" subsegment of the XRI "xri://=example*home*base/foo":

Comment [DSR8]: Reminder to discuss this in detail with Gabe.

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

<XRDS xmlns="...">
<XRD>
  <Query>*home</Query>
  <ProviderID>
    urn:uuid:2BA56CDE-9438-11D9-8BDE-F66BAD1E3F3A
  </ProviderID>
  <Synonym xref="true">xri://@example2/path</Synonym>
   …
</XRD>
</XRDS>
```

The resulting XRI would be "xri://@example2/path/*base/foo". Unless the client application has specific reason to believe otherwise, this is an error.

Note that for lookahead resolution, the resolving server MUST pass back to the client the chain of XRDs that ends in the XRD containing the XRI redirect. Because the use of an XRI redirect may be affected by policy or stateful information in the client, resolution servers cannot determine or express how an XRI redirect should be interpreted. Clients MUST re-initiate resolution for XRIs constructed as a result of applying the redirect mechanism defined in this section.

3.3 IRI Authority Resolution

From the standpoint of generic authority resolution, an IRI authority segment represents either a DNS name or an IP address at which an XRD for the authority may be retrieved. Thus IRI authority resolution simply involves making an HTTP(S) GET request to a URI constructed from from the IRI authority segment. The resulting XRD can then be used in the same manner as one obtained using XRI authority resolution.

While the use of IRI authorities provides backwards compatibility with the large installed base of DNS- and IP-identifiable resources, IRI authorities do not support the additional layer of abstraction, delegation, and extensibility offered by XRI authority syntax. Therefore IRI authorities are not recommended for new deployments of XRI identifiers.

This section defines IRI authority resolution as simple extension to the XRI authority resolution protocol defined in the preceeding section.

3.3.1 Service Type

Because IRI authority resolution takes place at a level “below” XRI authority resolution, it cannot be described in an XRD, and thus there is no corresponding resolution service type defined in section [2.2].

3.3.2 Protocol

Following are the normative requirements for IRI authority resolution that differ from generic XRI authority resolution:

1. The next resolution URI is constructed by extracting the entire IRI authority segment and prepending the string “http://”. See the exception in section [3.3.4].
2. The HTTP GET request MUST include an HTTP Accept header containing only the following:

```
Accept: application/xrd+xml
```

3. The HTTP GET request MUST have a Host: header (as defined in section 14.23 of **[RFC2616]**) containing the value of the IRI authority segment.
4. An HTTP server acting as an IRI authority SHOULD respond with the XRD for that authority.
5. The responding server MUST use the value of the Host header to populate the `xrd:XRD/xrd:Query` element in the resulting XRD. For example:

```
Host: example.com
```

3.3.3 Lookahead Resolution

Because IRI authority resolution is required to process the entire IRI authority segment in a single step, lookahead resolution does not apply.

3.3.4 Optional Use of HTTPS

Section [4] of this specification defines trusted resolution only for XRI authorities. Trusted resolution is not defined for IRI Authorities. If, however, an IRI authority is known to respond to HTTPS requests (by some means outside the scope of this specification), then the resolving client MAY use HTTPS as the access protocol for retrieving the authority’s XRD. If the resolving client is satisfied, via transport level security mechanisms, that the response is from the expected IRI authority, the resolving client may place greater trust in the contents of this XRD.

4 Trusted Authority Resolution

4.1 Introduction

This section defines a service for performing trusted XRI authority resolution as an extension of the generic XRI authority resolution service defined in the previous section.

In trusted resolution, the client requests a content type of "`application/xrd-saml+xml`", and the resolution server responds with an XRD that contains an additional element—a digitally signed SAML **[SAML]** assertion that asserts the validity of the containing XRD. If the response does not contain a valid, digitally signed SAML assertion as defined in section [4.5], this is an error and trusted resolution must not proceed.

This trusted resolution protocol does not provide a means to encrypt the contents of resolution requests and responses, nor does it provide a means for a responder to provide different responses for different requestors. These services may be provided by other security protocols used in conjunction with this specification, but confidentiality and client-authentication are explicitly out of scope of this version of this specification.

4.2 Service Type

The following sections define the resolution service with the `xrd:XRD/xrd:Service/xrd:Type` element value "xri://$resolution*authority*trusted*($v*2.0)". See section [2.2].

4.3 Protocol

This section normatively defines client and server behavior in trusted authority resolution.

4.3.1 Client Requirements

For a resolution client, trusted resolution is identical to the generic resolution protocol (section [3]) with the addition of the following requirements:

- The client SHOULD NOT request trusted resolution service from an authority unless an XRD describing the authority contains an `xrd:Descriptor/xrd:Service/xrd:Type` element with one of the following values:

  `xri://$resolution*authority*trusted*($v*2.0)`

  `xri://$resolution*authority*trusted`

- In a resolution request, the client MUST include an HTTP Accept header with the media type identifier "`application/xrd-saml+xml`". Clients willing to accept either trusted or untrusted resolution descriptors may use a combination of "`application/xrd-saml+xml`" and "`application/xrd+xml`" in the Accept header as described in section 14.1 of **[RFC2616]**. Media type identifiers SHOULD be ordered according to the client's preference for the media type of the response.
- A client MAY request lookahead resolution of multiple subsegments as defined in section [4.4].
- The client MUST individually validate each XRD in a resolution chain according to the rules defined in section [4.5]. When `xrd:XRD` elements come both from freshly-retrieved XRD documents and from a local cache, a client MUST ensure that these requirements are satisfied each time a resolution request is performed.

4.3.2 Server Requirements

For a resolution server, trusted resolution is identical to the generic resolution protocol (section [3]) with the addition of the following requirements:

- The HTTP(S) response to a trusted resolution request MUST include a content type of "`application/xrd-saml+xml`".
- The XRD returned by the server MUST contain a `saml:Assertion` element as an immediate child of `xrd:XRD` that is valid per the processing rules described by **[SAML]**.
- The SAML assertion MUST contain a valid enveloped digital signature as defined by **[XMLDSig]** and as constrained by section 5.4 of **[SAML]**.
- The signature MUST apply to the `xrd:XRD` element that contains the signed SAML assertion. Specifically, the signature MUST contain a single `ds:SignedInfo/ds:Reference` element, and the `URI` attribute of this reference MUST refer to the `xrd:XRD` element that is the immediate parent of the signed SAML assertion. The URI reference MUST NOT be empty; it MUST refer to the identifier contained in the `xrd:id` attribute of the `xrd:XRD` element.
- In **[SAML]**, the digital signature enveloped by the SAML assertion may contain a `ds:KeyInfo` element. If this element is included, it MUST describe the key used to verify the digital signature element. Because the signing key is known in advance by the resolution client, the `ds:KeyInfo` element SHOULD be omitted from the digital signature.
- The `xrd:XRD/xrd:Query` element MUST be present, and the value of this field MUST match the XRI Authority subsegment requested by the client.
- The `xrd:XRD/xrd:ProviderID` element MUST be present and its value MUST match the value of the `xrd:XRD/xrd:Service/xrd:ProviderID` element in an XRD advertising availability of trusted resolution service from this authority as required in section [4.6].
- The `xrd:XRD/saml:Subject/saml:NameID` element MUST be present and equal to the `xrd:XRD/xrd:Query` element.
- The `NameQualifier` attribute of the `xrd:XRD/saml:Assertion/saml:Subject/saml:NameID` element MUST be present and MUST be equal to the `xrd:XRD/xrd:ProviderID` element.
- There MUST be exactly one `saml:AttributeStatement` present in the `xrd:XRD/saml:Assertion` element. It MUST contain exactly one `saml:Attribute` element with a `Name` attribute of "xri://$xrd*($v*2.0)". This `saml:Attribute` element MUST contain exactly one `saml:AttributeValue` element whose text value is a URI reference to the `xrd:id` attribute of the `xrd:XRD` that is an immediate parent of the `saml:Assertion` element.

4.4 Lookahead Resolution

If a resolving client requests trusted resolution of multiple authority subsegments (see section [3.2.4]), the responding server SHOULD attempt to perform trusted resolution on behalf of the client as described in this section. However if the server is not able to obtain trusted XRDs for one or more additional lookahead subsegments, it SHOULD return only the trusted XRDs it has obtained and allow the client to continue.

4.5 Client Validation of XRDs

For each XRD returned as part of a trusted resolution request, the client MUST validate the XRD according to the rules defined in this section.

- The `xrd:XRD/saml:Assertion` element MUST be present.
- This assertion MUST valid per the processing rules described by **[SAML]**.
- The `saml:Assertion` MUST contain a valid enveloped digital signature as defined by **[XMLDSig]** and constrained by Section 5.4 of **[SAML]**.
- The signature MUST apply to the `xrd:XRD` element containing the signed SAML assertion. Specifically, the signature MUST contain a single `ds:SignedInfo/ds:Reference` element, and the `URI` attribute of this reference MUST refer to the `xrd:id` attribute of the `xrd:XRD` element that is the immediate parent of the signed SAML assertion.
- If the digital signature enveloped by the SAML assertion contains a `ds:KeyInfo` element, the client MAY reject the signature if this key does not match the signer's expected key as specified by the `ds:KeyInfo` element present in the XRD Descriptor that was used to describe the current authority. See section [4.6].
- The value of the `xrd:XRD/xrd:Query` element MUST matches the subsegment whose resolution resulted in the current XRD.
- The value of the `xrd:XRD/xrd:ProviderID` element MUST match the value of the `xrd:XRD/xrd:Service/xrd:ProviderID` element in any XRD advertising availability of trusted resolution service from this authority as required in section [4.6].
- The value of the `xrd:XRD/xrd:ProviderID` element MUST match the value of the `NameQualifier` attribute of the `xrd:XRD/saml:Assertion/saml:Subject/saml:NameID` element.
- The value of the `xrd:XRD/xrd:Query` element MUST match the value of the `xrd:XRD/saml:Assertion/saml:Subject/saml:NameID` element.
- There MUST exist exactly one `xrd:XRD/saml:Assertion/saml:AttributeStatment` with exactly one `saml:Attribute` element that has a `Name` attribute of "xri://$xrd*($v*2.0)". This `saml:Attribute` element must have exactly one `saml:AttributeValue` element whose text value is a URI reference to the `xrd:id` attribute of the `xrd:XRD` element that is the immediate parent of the signed SAML assertion.

If any of the above requirements are not met for an XRD in the trusted resolution chain, the result MUST NOT be considered a valid trusted resolution response as defined by this document. Note that this does not preclude a client from considering alternative resolution paths. For example, if an XRD advertising trusted resolution service has two or more `xrd:XRD/xrd:Service/xrd:URI` elements and the response from one service endpoint fails to meet the requirements above, the client MAY repeat the validation process using the second URI. If the second URI passes the tests, it MUST be considered a trusted resolution response as defined by this document and trusted resolution may continue.

4.6 Correlation of ProviderID and KeyInfo Elements

Each XRI authority participating in trusted authority resolution MUST be associated with at least one unique persistent service provider identifier expressed in the `xrd:XRD/xrd:Service/xrd:ProviderID` element of any XRD advertising trusted authority resolution service. This ProviderID value MUST NOT ever be reassigned to another XRI authority. A ProviderID may be any valid URI that meets these requirements of persistence and

uniqueness. Examples of appropriate URIs include URNs as defined by **[RFC2141]** and fully persistent XRIs converted to URI-Normal Form as defined by **[XRISyntax]**.

The purpose of ProviderIDs in XRI resolution is to enable XRI clients to correlate the metadata in an XRD advertising trusted authority resolution service with the response received from a trusted resolution server. If the signed XRD response contains the same ProviderID as the XRD used to advertise a service, and the client has reason to trust the signature, the client can trust that the XRD response has not been maliciously replaced with another XRD.

There is no defined discovery process for the ProviderID for a community root authority; it must be published in the community root XRD (or other equivalent description document—see section [2.5]) and verified independently. Once the community root XRD is known, the ProviderID for XRI authorties within this community MAY be discovered using the `xrd:XRD/xrd:Service/xrd:ProviderID` element for all resolution services offer by an authority. This trust mechanism may also be used for other services offered by this authority.

In addition, the metadata necessary for trusted authority resolution or other SAML **[SAML]** interactions MAY be discovered using the `ds:KeyInfo` element (section [2.3].) Again, if this element is present in the XRD advertising trusted authority resolution service (or any other service), and the client has reason to trust this XRD, the client MAY use the associated ProviderID to correlate the contents of this element with a signed response.

To assist clients in using this key discovery mechanism, it is important that servers be configured to sign responses in such a way that the signature can be verified using the correlated `ds:KeyInfo` element. For more information, see **[SAML]**.

5 Local Resolution

5.1 Introduction

This section defines a generic service for performing resolution of the local part of an XRI (anything that follows the authority segment) as an extension of the generic XRI authority resolution service defined in section [3]. Whereas authority resolution produces only one representation of a resource—an XRD—local resolution may be used to request either an XRD or any other representation of a resource. If an XRD is requested, it may include any of the metadata defined in an XRD (section [2.1]), or extensions to an XRD (section [2.7]). Requests for other representations may use the full HTTP semantics as defined in **[RFC2616]**.

Note that this is only one local resolution service. Other local resolution services could be defined—for example, an LDAP or DSML local resolution protocol that specified 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 additional protocols is left to future specifications.

New local resolution service types intended for widespread use SHOULD be identified with XRIs in the "$resolution*local" namespace. This namespace may be extended by any authority using cross-references as described in section 2.2.2 of **[XRISyntax]**. Local resolution service types defined solely for use within a private or closed community MAY have service types identified by any XRI, IRI, or URI.

5.2 Service Type

The following sections define the resolution service with the `xrd:XRD/xrd:Service/xrd:Type` element value "xri://$resolution*local*generic*($v*2.0)". See section [2.1].

5.3 Protocol

Following are the normative requirements for generic local resolution that differ from generic authority resolution:

1. There is no explicit form of lookahead resolution. See section [5.4].
2. The next resolution URI is constructed as defined in section [5.5].
3. If the client includes an Accept header with the value of "`application/xrd+xml`", the responding server MUST return an XRD, or a 404 error if no XRD is available to describe the identified resource.
4. If the client desires a different content type, it MAY select from the values of any `xrd:XRD/xrd:Service/xrd:MediaType` child elements of an `xrd:XRD/xrd:Service` advertising local resolution service. The client MAY also request any other Accept content type value.
5. If the client does not include an Accept header or its value is null, the responding server MAY return the resource representation of its choice, or a 3XX redirect to another representation of the resource.
6. A successful response that constitutes an XRI redirect as defined in section [3.2.7] SHOULD be processed as defined in that section. In other words, it is an instruction from the local resolution server to restart resolution with a new XRI.

5.4 Lookahead Resolution

Because generic local resolution processes the entire local part of an XRI in a single step, there is no explicit definition of lookahead resolution. However implicit lookahead resolution is possible, i.e. a local resolution server may follow local community policies to process a local resolution request consisting of multiple path segments and/or a query component by requesting resolution of path segments, subsegments, or query components from other servers.

5.5 Construction of the Next Resolution URI

For an XRI resolver, construction of the next resolution URI for local resolution is similar to construction of the next resolution URI for authority resolution as defined in section [3.2.5], except it involves additional rules based on the local part of the XRI as described in this section.

First, the XRI resolver selects the resolution endpoint URI from the current XRD (the XRD describing the authority for the local part of the XRI) according to the following rules:

1. Select the highest priority `xrd:XRD/xrd:Service` element whose child `xrd:XRD/xrd:Service/xrd:Pattern` element value matches the local part of the XRI as defined in section [5.6].
2. If step 1 fails, select the highest priority `xrd:XRD/xrd:Service` element whose child `xrd:XRD/xrd:Service/xrd:Type` element value matches the following string:

   `xri://$resolution*local*generic*($v*2.0)`
3. If step 2 fails, repeat step 2 for the following strings in the order they are listed until a match is found (if no match is found, return [ref to correct error message for this condition]:

   `xri://$resolution*local*generic`

   `xri://$resolution*local`

   `xri://$resolution`
4. For the selected Service element, select the highest priority URI element that contains an HTTP or HTTPS URI (depending on the desired transport protocol.) This is the resolution endpoint URI.
5. If no matching Service element is found, return a 404.

Comment [DSR9]: Same comment as earlier - do we need this step?

If the resolution endpoint URI ends with a forward slash (“/”), it MUST be removed before proceeding.

Second, in order for the local resolution service to understand the full context of the local part, the query XRI consists of the entire XRI currently being resolved. This query XRI MUST be in URI-normal form as required in section [2.2].

The final step is for the XRI resolver to append the query XRI to the service endpoint URI as a query string, i.e., following the question mark (“?”) character. The resulting URI is the next resolution URI.

Construction of the next resolution URI for local resolution services is more formally described in the following pseudo-code:

```
if (local-res-uri ends in “/”):
   remove trailing “/” in local-res-uri

local-res-uri = local-res-uri + “?” + uri-escape(XRI)
```

5.6 Selecting a Service Endpoint Using a Pattern

The `xrd:Service/xrd:Pattern` element enables XRI authorities to control the selection of a service endpoint using the value of the local part of the XRI. For example, the authority for

1020 “xri://=example” may desire XRI resolvers to select one service endpoint for the XRI
1021 “xri://=example/(+contact)” and a different service endpoint for the XRI
1022 “xri://=example/(+authentication)”.

1023 The value of the `xrd:Service/xrd:Pattern` element is a string representing a regular
1024 expression. The rules for selecting a service endpoint using this regular expression are:

1025 1. For all `xrd:XRD/xrd:Service/xrd:Pattern` elements in the XRD, select the set
1026 where the local part of the XRI matches the regular expression.
1027 2. If the result of step 1 is a single element, select its parent `xrd:XRD/xrd:Service`
1028 element.
1029 3. If the result of step 1 contains more than one element, select the highest priority parent
1030 `xrd:XRD/xrd:Service` element.
1031 4. If the result of step 1 contains no elements, there is no matching pattern and selection of
1032 a Service element should proceed as defined elsewhere in this specification.

1033

6 Proxy Resolution

6.1 Introduction

This section defines a service for performing HTTP(S) proxy resolution of an entire XRI (both the authority segment and the local part) as an extension of the generic XRI authority resolution protocol defined in section [3] and the generic local resolution protocol defined in section [5]. It also defines a standard syntax for expressing an XRI as an HTTP XRI, called an *HXRI*.

While proxy resolution may be used to request an XRD or any other representation of a resource, it is primarily intended to return an HTTP(S) redirect to clients such as browsers that have no native understanding of XRIs or XRI resolution but can process HXRIs. This provides backwards compatability with the large installed base of existing HTTP clients and servers.

6.2 Service Type

The following sections define the resolution service with the `xrd:XRD/xrd:Service/xrd:Type` element value “xri://$resolution*proxy*($v*2.0)”. See section [2.1].

It may appear to be of limited value to advertise proxy resolution service in an XRD to clients who need proxy resolution service because they not understand XRDs. However advertising a proxy resolution service, particularly in the XRD for a community root authority, can be useful to applications that need to automatically generate HXRIs for resolution by a community proxy resolver. Those applications can obtain the current HXRI prefix for proxy resolution from the community root XRD. See sections [2.6] and [6.4].

6.3 Protocol

Following are the normative requirements for proxy resolution that differ from generic authority resolution or generic local resolution:

1. The query XRI (QXRI) MUST be a valid XRI as specified in section [6.4].
2. The responding proxy resolution server MUST first perform lookahead authority resolution as defined in section [6.5].
3. If the QXRI does not include a local part, further processing MUST follow the rules in section [6.6].
4. If the QXRI includes a local part, further processing MUST follow the rules in section [6.7].
5. To provide a 3XX redirect, a proxy resolution server MUST select the redirect URI by the rules defined in section [6.8].
6. Proxy resolution servers SHOULD support the special processing instructions defined in section [6.9].
7. If, during local resolution, a proxy resolution server receives an XRI redirect as defined in section [3.2.7], it MUST be processed as defined in section [3.2.7]. In other words, resolution should continue with the highest priority cross-reference synonym.
8. Proxy resolvers are uniquely positioned to take advantage of caching and SHOULD use it to resolve the same XRIs or XRI components for multiple clients as defined in section [7.2.1].

6.4 HXRIs and QXRIs

Definition of a standard syntax for expressing an XRI as an HTTP XRI (HXRI) has two benefits:

- It allows XRIs to be used anyplace an HTTP URI can appear, including in Web pages, electronic documents, email messages, instant messages, etc.
- It allows XRI-aware processors and search agents to recognize an HXRI and extract the embedded XRI for direct processing and indexing.

To make this syntax as simple as possible for XRI-aware processors or search agents to recognize, an HXRI consists of a fully qualified HTTP or HTTPS URI authority segment that begins with the domain name "xri.". The query XRI (QXRI) is then appended as the entire local path. In essence, the proxy resolver domain name plus the terminating forward slash serves as a machine-readable prefix for an absolute XRI.

The normative ABNF for an HXRI is defined below based on the `XRI` and `ireg-name` productions defined in **[XRISyntax]**:

```
HXRI              = proxy-resolver "/" QXRI

proxy-resolver    = ( "http://" / "https://" ) proxy-reg-name

proxy-reg-name    = "xri." ireg-name

QXRI              = XRI
```

URI processors that recognize XRIs SHOULD interpret the local part of any HTTP or HTTPS URI that conforms to this ABNF as an XRI provided the domain name is at least three levels deep (e.g., "xri.example.com".) If a URI meets this test but the domain name is less than two levels deep, URI processors SHOULD interpret the local part as an XRI only if the domain has explicitly declared that it functions as an XRI proxy resolver.

Publishers of XRIs that need to be understood by non-XRI-aware clients SHOULD publish them as HTTP URIs conforming to the HXRI production.

6.5 Lookahead Resolution

Proxy resolution servers MUST perform lookahead resolution of the authority segment of the QXRI on behalf of their client as defined in section [3.2.4] ,with the addition of the following rules:

1. If the proxy resolution server receives an XRI redirect during authority resolution as defined in section [3.2.7], it MUST be processed as defined in section [3.2.7]. In other words, resolution should continue with the highest priority cross-reference synonym.
2. If lookahead authority resolution fails and the proxy resolution request includes an Accept header with the content type value of "`application/xrd+xml`" or "`application/xrd-saml+xml`", the responding proxy resolution server SHOULD return an error as defined in section [7.1].
3. If lookahead authority resolution fails and the proxy resolution request does not include an Accept header, or the Accept header has any other content type value, the proxy resolution server SHOULD return a 404.

6.6 Processing Rules Without a Local Part

If the QXRI does not include a local part, further processing is determined by the value of the Accept content type in the proxy resolution request according to the following rules.

1. A proxy resolution server MAY support requests for the content types "`application/xrd+xml`" and/or "`application/xrd-saml+xml`". If the client includes an Accept header specifying one of these content types, the proxy resolution

server MUST return the complete chain of XRDs for the authority segment, including any errors as defined in sections [3] or [4]. The chain of XRDs MUST begin with the XRD for the community root authority. If the community root authority does not publish an XRD itself, the proxy resolution server MUST construct one from the equivalent information published by the community root authority. See section [2.6].

2. If the proxy resolution request includes an Accept header with another content type value, the proxy resolution server MUST return either: a) a representation of the identified resource conforming to that content type (if it has access), b) a 3XX redirect as specified in section [6.8], or c) a 404 if neither is available.
3. If the client does not include an Accept header or its content type value is null, the proxy resolution server MUST return either: a) a representation of the described resource OTHER than an XRD (if it has access), b) a 3XX redirect as specified in section [6.8], or c) a 404 if neither is available.

6.7 Processing Rules With a Local Part

If a QXRI contains a local part, further processing is determined by the value of the Accept content type in the proxy resolution request according to the following rules:

1. If the proxy resolution request includes an Accept header with the content type value of "`application/xrd+xml`" or "`application/xrd-saml+xml`", the responding proxy resolution server MAY also perform local resolution to obtain an XRD as defined in section [5]. If the proxy resolution server elects to perform this step, it MUST include this XRD in the complete chain of XRDs it returns as specified in section [6.6].
2. If the proxy resolution request includes an Accept header with any other content type value, the proxy resolution server MUST either: a) perform local resolution as defined in section [5] to locate an XRD with a service endpoint advertising one of these content type values using a `xrd:Service/xrd:MediaType` element, b) return a 3XX redirect as specified in section [6.8], or c) return a 404 if neither is available.
3. If the proxy resolution request does not include an Accept header or its value is null, the proxy resolution server MUST either: a) perform local resolution as defined in section [5] to obtain a redirect URI (section [6.8]), b) return a 3XX redirect as specified in section [6.8], or c) return a 404 if neither is available.

6.8 Selection of a Redirect URI

If a proxy resolution server does not have access to a resource representation and must return a 3XX redirect, it MUST select the redirect URI according to the rules in this section.

6.8.1 Without a Local Part

If a QXRI consists of an authority segment with no local part, the redirect URI is selected as follows:

1. Select the highest priority `xrd:XRD/xrd:Service` element whose child `xrd:XRD/xrd:Service/xrd:Type` element value matches the following string:

```
xri://$resolution*uri
```

2. If step 1 succeeds, select the contents of the highest priority `xrd:XRD/xrd:Service/xrd:URI` child element. This is the redirect URI.
3. If step 1 fails, the authority producing the XRD is not authoritative for the redirect URI, and the proxy resolution server MUST either: a) perform local resolution to obtain an authoritative XRD as defined in section [5], or b) redirect to the next resolution URI for a local resolution service constructed as defined in section [5.5].

4. If the proxy resolution server elects to perform step 3a above, and the result is NOT an XRI redirect (see section [3.2.7]), then the steps in this section MUST be repeated on the XRD returned by the local resolution service.
5. If step 3 fails (there is no local resolution service), or if step 4 fails (there is no match), no redirect URI is available and the proxy resolution server SHOULD return a 404.

6.8.2 Local Part Matching a Pattern

If the query XRI includes a local part, the redirect URI is selected according to the following rules:

1. Select a `xrd:XRD/xrd:Service` element using pattern matching according to the rules defined in section [5.6].
2. If step 1 fails, proceed to section [6.8.3].
3. If step 1 succeeds, compare the value of the `xrd:XRD/xrd:Service/xrd:Type` child element to the following string:

   ```
   xri://$resolution*uri
   ```

4. If step 3 is a match, proceed with step 3 in section [6.8.3].
5. If step 3 is not a match, select the contents of the highest priority `xrd:XRD/xrd:Service/xrd:URI` child element. This is the base URI.
6. If the base URI ends in a forward slash, construct the redirect URI by removing the forward slash and appending the entire QXRI as a query string as defined in section [5.5].

6.8.3 Local Part Not Matching a Pattern

If the query XRI includes a local part but there is no pattern match in section [6.8.2], the redirect URI is selected according to the following rules:

1. Select the highest priority `xrd:XRD/xrd:Service` element whose child `xrd:XRD/xrd:Service/xrd:Type` element value matches the following string:

   ```
   xri://$resolution*uri
   ```

2. If step 1 succeeds, select the contents of the highest priority `xrd:XRD/xrd:Service/xrd:URI` child element.
3. If this URI ends in a forward slash, remove the forward slash, and append the entire local part of the QXRI to the path. This is the redirect URI.
4. If step 1 fails, the authority producing the XRD is not authoritative for the redirect URI, and the proxy resolution server MUST either: a) perform local resolution to obtain an authoritative XRD as defined in section [5], or b) redirect to the next resolution URI for a local resolution service constructed as defined in section [5.5].
5. If the proxy resolution server elects to perform step 4a above, and the result is NOT an XRI redirect (see section [3.2.7]), then the steps in this section MUST be repeated on the XRD returned by the local resolution service.
6. If step 4 fails (there is no local resolution service), or if step 5 fails (there is no match), no redirect URI is available and the proxy resolution server SHOULD return a 404.

6.9 Special Processing Instructions

Since a proxy resolution server acts as an XRI client with a HTTP URI interface, it can be useful to application developers and network administrators to pass special processing instructions in an HXRI. Table Table 7 defines two such processing instructions that MAY be passed in the QXRI.

Processing instruction	Purpose	Parameter

$resolution*xrd	Instructs the proxy resolution server to return an XRD even though no Accept content type is specified	QXRI appended as the path after the processing instruction
$resolution*uri	Instructs the proxy resolution server to return a redirect URI in the BODY of the response rather than as a 3XX redirect	QXRI appended as the path after the processing instruction

Table 7: Special processing instructions for proxy resolvers.

A proxy resolution service conformant with this specification SHOULD accept these processing instructions. It MAY also accept and respond to other processing instructions defined in other specifications or locally.

Figure 1 shows examples of HXRIs including these processing instructions for a proxy resolver located at “http://xri.example.org”.

```
http://xri.example.org/$resolution*xrd/=example*foo/bar
http://xri.example.org/$resolution*uri/=example*foo/bar
```

Figure 1: Examples of HXRIs including processing instructions.

6.10 Differences Between Proxy Resolution Servers

A proxy resolution request may be sent to any proxy resolution server that will accept the request. All proxy resolution servers SHOULD attempt to deliver uniform responses given the same QXRI. However because proxy resolution servers are required to perform XRI resolution as a client, they must make decisions about priorities, XRI redirects, and trust policies on behalf of the client they are proxying. Since they decisions may be based on local policy or implementation, it is possible for different proxy resolution servers to return different results given the same QXRI.

7 Use of HTTP(S)

7.1 HTTP Errors

Proxy and lookahead resolvers MUST "pass through" to the resolving client any HTTP error codes resulting from resolution if the proxy or lookahead resolver cannot proceed with resolution due to an HTTP error condition. For example, if, during resolution on behalf of a client, a proxy resolution server is returned a 404 error code by another authoritative resolution server, it must return that 404 code to its client.

Upon encountering an HTTP error code that halts the proxy or lookahead resolver's ability to complete resolution, the proxy or lookahead resolver MUST return an `xrd:XRDS` document in the body of the HTTP error response. This `xrd:XRDS` document MUST contain the list of `xrd:XRD` elements corresponding to the subsegments successfully resolved or retrieved from cache. For example, if a proxy is asked to resolve @a*b*c, and successfully resolves @a*b, but receives a HTTP 404 on resolving *c, it will return an HTTP 404 response to its client that include `xrd:XRD` elements for @, *a, and *b. In this way, the resolver indicates to the resolving client that *c is the subsegment causing the 404 response.

This use of error codes, while slightly unusual, conforms to the requirements of **[RFC2616]**, specifically sections 10.4 and 10.5, which state that "the server SHOULD include an entity containing an explanation of the error situation." The combination of the error code and the list of successfully resolved `xrd:XRD` elements explains to the client exactly which subsegment caused the error. This should save both the client and the authority returning the error code an extra HTTP request/response cycle.

Even when given an HTTP error response, resolving clients SHOULD consider the `xrd:XRD` elements returned in the content of the HTTP response as valid cacheable responses. All other rules about XRDs, including those specified in Section 4 for trusted resolution, also apply.

7.2 HTTP Headers

7.2.1 Caching

The HTTP caching capabilities described by **[RFC2616]** should be leveraged for all types of XRI resolution service. Specifically, implementations SHOULD implement the caching model described in section 13 of **[RFC2616]**, and in particular, the "Expiration Model" of section 13.2, as this requires the fewest round-trip network connections.

All XRI resolution servers SHOULD send the Cache-Control or Expires headers in their responses per section 13.2 of **[RFC2616]** unless there are overriding security or policy reasons to omit them.

Note that HTTP Cache headers SHOULD NOT conflict with expiration information in an XRD. That is, the expiration date specified by HTTP caching headers SHOULD NOT be later than any of the expiration dates for any of the `xrd:XRD/xrd:Expires` elements returned in the HTTP response. This implies that lookahead and proxy resolvers SHOULD compute the "soonest" expiration date for the XRDs in a resolution chain and ensure a later date is not specified by the HTTP caching headers for the HTTP response.

7.2.2 Location

During HTTP interaction, "Location" headers may be present per **[RFC2616]** (i.e., during 3XX redirects). Redirects SHOULD be made cacheable through appropriate HTTP headers, as specified in section 7.2.1.

7.2.3 Content-Type

For authority resolution, the “Content-type” header in the 2XX responses MUST contain the value “`application/xrd+xml`” or “`application/xrd-saml+xml`” specifying that the content is a generic XRD (section [3]) or a trusted XRD (section [4]) respectively.

For local resolution, clients and servers MAY negotiate content type using standard HTTP content negotiation features. Regardless of whether this feature is used, however, the server MUST respond with an appropriate media type in the “Content-type” header if the resource is found and an appropriate content type is returned.

To obtain a redirect in proxy resolution, clients MUST NOT use the media types defined for authority resolution.

7.3 Other HTTP Features

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

7.4 Caching and Efficiency

In addition to HTTP-level caching, resolution clients are encouraged to perform caching at the application 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 SHOULD NOT use cacheable redirects with expiration times sooner than the expiration times of other HTTP responses in the resolution chain. In general, all XRI deployments should be mindful of limitations in current HTTP clients and proxies.

The cache expiration time of an XRD may also be explicitly limited by the identifier authority. If the expiration time in `xrd:XRD/xrd:Expires` is sooner than the expiration time calculated from the HTTP caching semantics, the XRD MUST be discarded before the expiration time in `xrd:XRD/xrd:Expires`. Note also that the SAML assertion used in trusted resolution (section [4]) may cause invalidation of a XRD even before HTTP caching semantics or the `xrd:XRD/xrd:Expires` element.

With both application-level and HTTP-level caching, the resolution process is designed to have minimal overhead. Resolution of each qualified subsegment of an XRI authority segment is a separate step described by a separate XRD, so intermediate results can typically be cached in their entirety. For this reason, resolution of higher-level (i.e., further to the left) qualified subsegments, which are common to more identifiers, will naturally result in a greater number of cache hits than resolution of lower-level subsegments.

8 Security and Data Protection

Significant portions of this specification deal directly with security issues, and these will not be summarized again here. In addition, basic security practices and typical risks in resolution protocols are well-documented in many other specifications. Only security considerations directly relevant to XRI resolution are included here.

8.1 DNS Spoofing

As the specified resolution mechanism is dependent on DNS, the accuracy of the XRI resolution response is dependent on the accuracy of the original DNS query. When trustworthy, unambiguous, authoritative responses are required, the trusted resolution protocol defined by this specification is recommended. Resolution results obtained using this protocol can be evaluated independently of DNS resolution results. While this does not solve the problem of DNS spoofing, it does allow the client to detect an error condition and reject the resolution result as untrustworthy. For environments that require higher confidence in the result of DNS resolution, DNSSEC **[DNSSEC]** is recommended as a supplement to trusted resolution as defined by this specification.

8.2 HTTP Security

Many of the security considerations set forth in HTTP/1.1 **[RFC2616]** apply to XRI Resolution protocols defined here. In particular, confidentiality of the communication channel is not guaranteed by HTTP. Server-authenticated HTTPS should be considered in cases where confidentiality of resolution requests and responses is desired.

Special consideration should be given to proxy and caching behaviors to ensure accurate and reliable responses from resolution requests. For various reasons, network topologies increasingly have transparent proxies, some of which may insert VIA and other headers as a consequence, or may even cache content without regard to caching policies set by a resource's HTTP authority.

Implementations of XRI Proxies and caching authorities should also take special note of the security recommendations in HTTP/1.1 **[RFC2616]** section 15.7

8.3 Caching Authorities

In addition to traditional HTTP caching proxies, XRI resolution authority proxies may be a part of the resolution topology. Such proxies should take special precautions against cache poisoning, as these caching entities may represent trust decision points within a deployment's resolution architecture.

8.4 Lookahead and Proxy Resolution

During proxy resolution, some or all of an XRI is provided to the proxy resolver. During lookahead resolution, subsegments of the XRI authority segment for which the resolving network endpoint is not authoritative may be revealed to that service endpoint.

In both cases, privacy considerations should be evaluated before disclosing such information.

8.5 SAML Considerations

Trusted resolution must adhere to the rules defined by the SAML 2.0 Core Specification. Particularly noteworthy are the XML Transform restrictions on XML Signature defined in SAML and the enforcement of the SAML Conditions element regarding the validity period.

8.6 Community Root Authorities

The XRI authority information for a community root needs to be well-known to the clients that request resolution within that community. For trusted resolution, this includes the URIs, the `xrd:XRD/xrd:ProviderID`, and the `ds:KeyInfo` information. An acceptable means of providing this information is for the community root authority to produce a self-signed XRD and publish it to a server-authenticated HTTPS endpoint. Special care should be taken to ensure the correctness of such an XRD; if this information is incorrect, an attacker may be able to convince a client of an incorrect result during trusted resolution.

8.7 Denial-Of-Service Attacks

XRI Resolution, including trusted resolution, is vulnerable to denial-of-service (DOS) attacks typical of systems relying on DNS and HTTP.

8.8 Limitations of Trusted Resolution

While the trusted resolution protocol specified in this document provides a way to verify the integrity of a successful XRI resolution, it does not provide a way to verify the integrity of a resolution failure. Reasons for this limitation include the prevalence of non-malicious network failures, the existence of denial-of-service attacks, and the ability of a man-in-the-middle attacker to modify HTTP responses when resolution is not performed over HTTPS.

Additionally, there is no revocation mechanism for the keys used in trusted resolution. Therefore, a signed resolution's validity period should be limited appropriately to mitigate the risk of an incorrect or invalid resolution.

9 References

9.1 Normative

[DNSSEC] D. Eastlake, *Domain Name System Security Extensions*, http://www.ietf.org/rfc/rfc2535, RFC 2535, March 1999.

[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.

[RFC2483] M. Meallling, R. Daniel Jr., *URI Resolution Services Necessary for URN Resolution,* http://www.ietf.org/rfc/rfc2483.txt, RFC 2483, January 1999.

[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.

[SAML] S. Cantor, J. Kemp, R. Philpott, E. Maler, *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0,* http://www.oasis-open.org/committees/security, March 2005.

[XMLDSig] D. Eastlake, J. Reagle, D. Solo et al., *XML-Signature Syntax and Processing*, World Wide Web Consortium, http://www.w3.org/TR/xmldsig-core/, February 12, 2002.

[XRIMetadata] D. Reed, *Extensible Resource Identifier (XRI) Metadata V2.0*, http://docs.oasis-open.org/xri/xri/V2.0/xri-metadata-V2.0-cd-01.pdf, March 2005.

[XRISyntax] D. Reed, D. McAlpin, *Extensible Resource Identifier (XRI) Syntax V2.0*, http://docs.oasis-open.org/xri/xri/V2.0/xri-syntax-V2.0-cd-01.pdf, March 2005.

9.2 Informative

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

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

[XRIGuide] [Editors here], XRI Implementer's Guide v2.0, [link here], Work-In-Progress, October 2005.

[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. XML Schema for XRDS and XRD (Normative)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="xri://$xrds" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xrds="xri://$xrds">
    <!-- Utility patterns -->
    <xs:attributeGroup name="otherattribute">
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:attributeGroup>
    <xs:group name="otherelement">
        <xs:choice>
          <xs:any namespace="##other" processContents="lax"/>
          <xs:any namespace="##local" processContents="lax"/>
        </xs:choice>
    </xs:group>
    <!-- Patterns for elements -->
    <xs:element name="XRDS">
        <xs:complexType>
          <xs:sequence>
             <xs:group ref="xrds:otherelement" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attributeGroup ref="xrds:otherattribute"/>
        </xs:complexType>
    </xs:element>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 4 U (http://www.xmlspy.com) by Drummond Reed (Cordance
Corporation) -->
<xs:schema targetNamespace="xri://$xrd*($v*2.0)" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xrd="xri://$xrd*($v*2.0)" xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <!-- Utility patterns -->
    <xs:attributeGroup name="otherattribute">
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:attributeGroup>
    <xs:group name="otherelement">
        <xs:choice>
          <xs:any namespace="##other" processContents="lax"/>
          <xs:any namespace="##local" processContents="lax"/>
        </xs:choice>
    </xs:group>
    <xs:complexType name="URIpattern">
        <xs:simpleContent>
          <xs:extension base="xs:anyURI">
             <xs:attributeGroup ref="xrd:otherattribute"/>
          </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="Stringpattern">
        <xs:simpleContent>
          <xs:extension base="xs:string">
             <xs:attributeGroup ref="xrd:otherattribute"/>
          </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!-- Patterns for elements -->
    <xs:element name="XRD">
        <xs:complexType>
          <xs:sequence>
             <xs:element ref="xrd:Query" minOccurs="0"/>
             <xs:element ref="xrd:Expires" minOccurs="0"/>
             <xs:element ref="xrd:Synonym" minOccurs="0" maxOccurs="unbounded"/>
             <xs:element ref="xrd:XSynonym" minOccurs="0" maxOccurs="unbounded"/>
```

```
                <xs:element ref="xrd:ProviderID"/>
                <xs:element ref="xrd:Service" minOccurs="0" maxOccurs="unbounded"/>
                <xs:group ref="xrd:otherelement" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute ref="xrd:id"/>
            <xs:attribute ref="xrd:version"/>
            <xs:attributeGroup ref="xrd:otherattribute"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Query" type="xrd:Stringpattern"/>
    <xs:element name="Expires">
        <xs:complexType>
            <xs:simpleContent>
                <xs:extension base="xs:dateTime">
                  <xs:attributeGroup ref="xrd:otherattribute"/>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="Synonym" type="xrd:URIpattern"/>
    <xs:element name="XSynonym" type="xrd:URIpattern"/>
    <xs:element name="ProviderID" type="xrd:URIpattern"/>
    <xs:element name="Service">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="xrd:Type" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element ref="xrd:MediaType" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element ref="xrd:Pattern" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element ref="xrd:URI" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element ref="xrd:ProviderID" minOccurs="0"/>
                <xs:group ref="xrd:otherelement" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attributeGroup ref="xrd:otherattribute"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Type" type="xrd:URIpattern"/>
    <xs:element name="MediaType" type="xrd:Stringpattern"/>
    <xs:element name="Pattern" type="xrd:Stringpattern"/>
    <xs:element name="URI" type="xrd:URIpattern"/>
    <xs:attribute name="priority" type="xs:nonNegativeInteger"/>
    <xs:attribute name="version" type="xs:string" fixed="2.0"/>
    <xs:attribute name="id" type="xs:ID"/>
</xs:schema>
```

Appendix B. RelaxNG Compact Syntax Schema for XRDS and XRD (Non-normative)

[Need new version in RelaxNG format.]

Appendix C. Acknowledgments

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

Comment [DSR10]: Needs updating.

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

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

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

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

Appendix D. 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 President.

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 President.