INTERNET-DRAFT
Date: March 13, 1998
Expires: September 18, 1998
draft-girod-w3-id-res-ext-00.txt

Lewis Girod
Benjie Chen
MIT Laboratory for Computer Science
Henrik Frystyk
World Wide Web Consortium
John Mallery
MIT Artificial Intelligence Laboratory

WIRE - W3 Identifier Resolution Extensions

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet- Drafts as reference material or to cite them other than as "work in progress."

To view the entire list of current Internet-Drafts, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), ftp.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Note: This is a very rough draft suitable only for experimental implementations. It is expected to change in the near future.

Abstract

WIRE extends HTTP with a new type of redirect response that permits a resolver to explicitly delegate a resolution to other resolvers and protocols. WIRE is an effort to make delegation more explicit, redirection more flexible, and resolution processes more efficient through the use of hints. This document defines WIRE and describes the expected behaviors of resolvers and clients using WIRE. WIRE is an extension of the HyperText Transfer Protocol (HTTP), and is intended to be compatible with HTTP/1.0 and above [4][5].

WIRE encourages use of long-lived URIs and at the same time supports protocol evolution without having to change currently deployed URIs or URI schemes. The extension is based on a simple URI resolution model that allows an application to dynamically request metadata describing where and how to access a resource. The model can use any generic metadata description language (e.g. RDF) and as the metadata itself is interpreted as a first class resource, metadata resources are no different than any other resource on the Web.

1 Introduction

1.1 Terminology

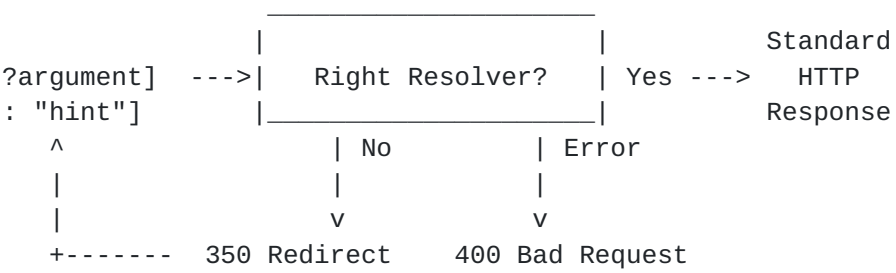
A resolver is an application that translates a URI into another URI or in case it is the authoritative resolver, directly to the requested resource.

A resolution process is the sequenced set of operations performed by a set of one or more resolvers is a nested set of operations that eventually will result in an entity being generated and returned to the requestor.

1.2 Resolution Model

URI resolution models have been a perennial source of confusion. In this section we present a new, clearer resolution model.

The WIRE model of URI resolution is fairly simple: to resolve a URI, ask the right server to resolve it, supplying any appropriate arguments or hints. The following diagram and discussion shows how this works for the "GET" method (other methods apply the same resolution process, differing only in the processing at the origin server):



If the location of the "right" resolver is unknown, the client sends a request to a favored or local resolver, optionally including a hint. This resolver will do some amount of work on the client's behalf, and will either return one of the following responses:

* A standard HTTP response, indicating that the resolver is authoritative for that URI and that resolution is complete
* A 350 redirect indicating that the resolution process is not done yet
* A 400 response, indicating that a failure occurred in the resolution infrastructure, and no further authoritative data can be obtained

A 200 response returns a view of the URI, or in the case of a query containing an argument, a view resulting from a method call on the resource identified by the URI. A 350 response returns hint information intended to help the client continue the resolution process. The hint information might describe resolvers that can further resolve the URI in question, or it might list alternate URIs that also resolve to the resource named by the original URI.

To understand this model it is important to have a broader understanding of

URI resolution. First, completion of resolution is largely in the eye of the beholder; the data one client resolves might be interpreted by another client as metadata that can be used to continue resolving the URI. In one sense a 350 redirect from an initial attempt at resolution is a view of the URI; in another sense it is simply metadata containing a hint for continued resolution at another site. When thinking about URI resolution it is important to keep this contextual information in mind. Second, the final step of resolution is performed by the resource itself, although in current implementations this activity is coordinated by the "server" process.

A very important semantic difference between the 350 response and other HTTP responses has to do with the trust model for a 350 response. Unlike other HTTP responses, the content of metadata contained within a 350 response is the responsibility of the resolver that serves it, and is not in any way linked to the origin server for the resource it describes. If the resolvers are not trusted there is no guarantee that the metadata is accurate. The efforts to specify signed RDF metadata should provide methods for correcting these deficiencies in the future.

1.3 Problems this model does NOT solve

Resolution using WIRE does not guarantee that a resolution process converges to a resolved resource. It does not guarantee that the resource returned is the correct or authoritative resource. It does not guarantee that a resolver understands every URI, nor that the authoritative resolver for any given URI exists or can be located. It does not guarantee that URIs are maintained in a persistent fashion or that URIs consistently resolve to resources that are conceptually equivalent. WIRE is designed to enable the construction of resolution systems that can provide some or all of these guarantees over specific namespaces.

1.4 Guide to this Document

This document is organized as follows: Section 2 describes the syntax of WIRE; Section 3 describes the semantics and behaviors of resolver, clients and proxies; Section 4 gives a short example; Section 5 discusses the merit of caching; and Section 6 discusses security issues.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 ([6]). In grammar definitions, this document uses the same parsing constructs as RFC 1630[1], RFC 2068 ([5]). In particular, the non-terminals uri is defined in [1][4][5].

2 Syntax of Extended Headers and Responses

This section introduces the headers and the response code defined by WIRE. This specification attempts to define minimal syntactical requirements for WIRE conformance.

2.1 Hints

URI resolution can be streamlined through the use of resolution hints. Resolution hints are encoded as URIs. Apart from the URI encoding rules, the syntax and semantics of resolution hints is specific to the resolution system indicated by the hint. For the purpose of caching and loop avoidance, two hints are lexically equivalent if they are octet-by-octet equal after applying URI normalization rules.

Resolution hints are intended to be processed by both clients and resolvers during the resolution process. To the client, hints convey resolver locations and supported protocols and type models. To the resolver indicated by the hint, hints convey client preferences and tokens important to the resolution process. For example, it is essential that URN resolvers receive a hint that includes a token describing a starting point for resolution. This allows the resolver to skip delegation steps that would otherwise be necessary to obtain the hint.

2.2 Extended Request Headers

2.2.1 Optional Header

The Optional request-header allows a client to declare itself to support WIRE. This header is optional. The presence of this header indicates that the resolver may return WIRE specific responses to the client. In the absence of this header the resolver must assume the client is not WIRE compliant. In this case the resolver may complete the resolution of the URI if it can do so without sending back WIRE specific responses, or it may reject the request with a 400 response code. Proxy behaviors are discussed in more detail in section 3.3. See [10] for a description of the use of the Optional header.

The Optional header has the following grammar:

```
Optional-Header          = "Optional" ":" WIRE-specification-URN
WIRE-specification-URN   = "urn:specs:WIRE/0.0"
```

2.2.2 Resolution-Hint header

The Resolution-Hint request-header can be used by a client to supply a resolution hint to the resolver. It has the following grammar:

```
Resolution-Hint-Header   = "Resolution-Hint" ":" hint
hint                     = quoted-absolute-uri
quoted-absolute-uri      = "\"" absolute-uri "\""
```

The Resolution-Hint header is optional. If this header is absent from a request, the resolver may either attempt to resolve the URI without a hint (in many cases this may slow down the resolution process) or reject the request with a 400 response code.

2.3 Extended Response Code - 350

In order to support distribution of resolution authority, WIRE includes the **350 response, a new response code intended to express delegation and** redirection in the resolution framework. To delegate or redirect the resolution of a URI, a WIRE resolver may return one or more alternate URIs, each bound to zero or more resolution hints. The Resolver-Location header in a 350 response encodes this comma delimited set of bindings. Within each binding, the list of hints is delimited by semicolons. The Resolver-Location header has the following grammar:

```
Resolver-Location-Header = "Resolver-Location" ":" binding *("," binding)
binding                  = alternate-uri *(";" hint)
alternate-uri            = "\"\"" | quoted-relative-uri | quoted-absolute-uri
hint                     = quoted-absolute-uri
quoted-relative-uri      = "\"" relative-uri "\""
```

As shown above, the alternate URI specified in the binding may appear as an absolute URI or in two other forms. If the binding applies to the original request-URI, the empty string ("") may be used instead of repeating the original request-URI. If a binding applies to a URI that can be expressed relative to the original request-URI, a relative URI may be quoted in place of an absolute URI.

A 350 response code may also include an entity containing additional metadata relevant to the request-URI. Note that the data in the Resolver-Location header and the optional entity in a 350 response follows the amended trust model described in section 1.

3 Semantics and Behavior of WIRE

3.1 Resolver Behaviors and Response Semantics

When a resolver receives a resolution request on a URI, the resolver should attempt to resolve the URI, making use of any supplied hint information. If the resolver is able to resolve the request-URI, the resolver processes the request as would a normal HTTP server.

If the resolver fails to make progress towards the resolution of the URI, a 4xx error code may be returned. The 404 error code should only be returned if the resolver has authority over the request-URI and that URI is not bound to a resource. Other 4xx codes indicate a temporary or permanent failure in the resolution infrastructure.

If the resolver makes progress towards resolution of the request-URI, a 350 response may be used to redirect resolution to alternate URIs or to delegate the resolution of the URI to alternate resolvers. Redirection and delegation information is conveyed via the Resolver-Location header defined in section 2.3**. The client may then attempt to use supplied hints to resolve any of the** alternate URIs listed in the header. The encoding and semantics of the hints is defined by individual resolution frameworks.

The hints supplied in a 350 response can specify a collection of alternate protocols or resolution systems. New types of hints, referring to new or

location-specific protocols, can be phased in along with more standard hints as fallback options. This can add scalability and flexibility to the resolution framework by providing information about alternate resolution mechanisms to clients that are aware of those alternate mechanisms.

A resolver may receive both standard HTTP requests and requests from WIRE-aware clients. Clients indicate awareness of WIRE by including an Optional request header with the URI of the WIRE specification. A WIRE-compliant resolver that receives a request without an appropriate Optional header may either reject the request with a 400 response code or proxy the resolution of the URI on behalf of the client. In the latter case, the resolver acts as a delegation proxy (section 3.3).

For caching purposes, a 350 response should also include information on the life-time of delegation/redirection responses. This information can be encoded in one of the HTTP caching headers ([4][5]).

3.2 Client Behaviors

When a client makes a resolution request to a resolver, the resolver cannot guarantee that the URI can be resolved. If the URI is within the authority of another resolver, a 350 response may be returned to the client. To continue the resolution process, the client should:

1. Parse the Resolver-Location header in the 350 response.
2. Select one of the alternate URIs, and select one of the resolvers specified by the hints bound to that URI.
3. Initiate a new request to the specified resolver, using the protocol and address information specified in the hint.
4. Request the selected URI, optionally including any appropriate hint information in the Resolution-Hint request header. Some resolvers require that the hint be forwarded while others do not.

Whenever new resolution hints are returned to a client, there is the possibility of a "delegation loop", in which a new hint is lexically equivalent to a hint previously applied in the resolution of the same request-URI. A "redirection loop" can occur if an alternate URI is lexically equivalent to a previous URI in a sequence of redirections. To detect delegation loops, a client should keep for each request-URI a history of resolution hints previously applied, and should lexically compare new resolution hints to the history. Detection of redirection loops should already be performed by web clients.

These methods do not guarantee the detection of delegation or redirection loops. Malicious resolvers can cause delegation loops by returning resolution hints that are semantically equivalent to a prior hint but are not lexically equivalent to any previous hint. Loop detection is only guaranteed to work for resolvers that recognize only a small set of distinct resolution hints.

Clients that wish to receive 350 responses must include the Optional header

with the URN of the WIRE specification in each resolution request. Older clients and clients that do not want to receive delegation responses may omit the Optional header.

3.3 Proxying WIRE Functionalities

WIRE extends the normal HTTP proxy behavior and additionally defines a new behavior called a delegation proxy. In both of these cases normal HTTP proxy semantics should be followed in addition to the WIRE specific semantics.

3.3.1 HTTP/WIRE Proxy Behavior

A WIRE resolver used as a proxy may proxy non-local resolution requests based on local policy decisions. If performing the resolution is contrary to policy, an appropriate error response should be returned. Otherwise, it should attempt to resolve the URI with or without a resolution hint, and the results of that effort to the client. If new resolution hint is supplied, the resolver should attempt to resolve the URI based on the text of the URI and global information. If the resolution hint supplied indicates a remote resolver, the proxy should initiate a request to that resolver.

3.3.2 Delegation Proxy Behavior

If the requesting client is not WIRE compliant, it cannot parse a 350 delegation/redirection response. In this case, a delegation proxy may choose to perform the entire resolution process on behalf of the client and pass the result of the resolution back to the original client. A delegation proxy should follow the guidelines for a WIRE client when performing the resolution. The first result entity that conforms to standard HTTP should be returned to the client.

A client requests delegation proxy behavior by omitting the Optional request header indicating WIRE. If a resolver's policy prohibits this behavior, or if at any time the resolver decides to abort the resolution, a 400 response should be returned to the client.

4 Example

A client wishes to resolve the URI urn:cid:9802032044@thebe.lcs.mit.edu. It sends a resolution request to http://urn.org/

```
GET urn:cid:9802032044@thebe.lcs.mit.edu HTTP/1.0
Host: urn.org
Optional: "urn:specs:WIRE/0.0"
```

The resolver at urn.org determines that the URI can be resolved using another resolver, and sends back

```
HTTP/1.0 350 Resolution Delegated
Resolver-Location: "";"res-hint:http://thebe.lcs.mit.edu/;scope=urn:cid:"
```

To continue the resolution process, the client makes another resolution

request, this time to http://thebe.lcs.mit.edu/

```
GET urn:cid:9802032044@thebe.lcs.mit.edu HTTP/1.0
Host: thebe.lcs.mit.edu
Optional: "urn:specs:WIRE/0.0"
Resolution-Hint: "res-hint:http://thebe.lcs.mit.edu/;scope=urn:cid:"
```

The resolver at thebe.lcs.mit.edu is the authoritative resolver for the URI. It returns the resolution result

```
HTTP/1.0 200 OK
<headers>

<entity>
```

5 Caching

The exact behaviors of URI resolution differ from resolution framework to resolution framework. For example, ignoring DNS lookups, URL resolutions are usually a one step process, whereas URN resolutions may take a few delegations to complete. We hope to realize significant performance gains for URI resolution frameworks that require more than one delegations through the caching of 350 responses. To cache a 350 response, both the request URI and the value of the Resolution-Hint header that was used in the request must be included in the key that maps to the cached entity. Standard HTTP headers that restrict or control caching must be heeded.

The need for caching 350 responses stems from the delegation nature of some URI resolution frameworks. Since the discovery process required to refresh a cached non-350 HTTP response may be significant, caching the 350 responses preceding the non-350 HTTP response may result in significant performance improvements, depending on cache expiration times.

One optimization when caching 350 responses might retain only the response that has resulted from the longest sequence of delegations, that is, discarding intermediate 350 responses. Applying this optimization, subsequent requests for the same URI may skip directly to the most specific delegation. If this optimization is applied, the most restrictive caching requirements for any response in the sequence must be applied to the most specific sequence.

A similar optimization would be to cache all the intermediate 350 responses, and for each subsequent request, re-use a non-expired delegation response with the longest sequence of delegations. Applying this optimization, most subsequent requests can skip many delegation steps.

6 Security Considerations

WIRE inherits the HTTP security model at the transport level. That does not, however, imply the safety and authenticity of the resolution metadata or resolved data. One possible approach to guarantee safety and authenticity is to include signed metadata. Those problems

require further study and are beyond the scope of this document.

Possible conflicts between the HTTP trust model and the 350 response raise security concerns, as mentioned in section 1. In short, 350 responses without security extensions are responses from untrusted resolvers. Measures such as loop-avoidance should be applied to detect and prevent denial-of-service attacks.

Implementations of WIRE should follow the security restrictions of the environment the resolver operates in. For example, Resolvers on firewalls operating under both single-step and delegation proxy behaviors may be required to filter out resolution requests from outside the firewall that intend to use an internal resource. Such requests, in most cases, are not allowed. However it is quite essential that such proxying resolvers forward resolution requests from internal clients to the outside world, unless an organization intend to mirror resolution services over all URI namespaces internally.

7 Acknowledgments

The motivation leading to this work stemmed from a few directions. John Mallery's experience with implementing the PDI namespace for URNs indicated that the THTTP ([8]) spec did not adequately cover error messages and redirection. Discussions with John Mallery and Henrik Frystyk Nielsen led to the initial formulation of this protocol specification, in an effort to rework THTTP into an official HTTP extension. Henrik Frystyk Nielsen also provided invaluable ideas and feedback on modifying the original design to fit into the generic ideas of URI resolution.

Karen Sollins and Dorothy Curtis have also provided many insightful ideas and feedback on the general resolution architecture and on this document.

8. Authors Addresses

Lewis Girod
Benjie Chen
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139, USA
Email: {girod,benjie}@lcs.mit.edu

Henrik Frystyk Nielsen
Technical Staff, World Wide Web Consortium
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139, USA
Email: frystyk@w3.org

John Mallery
MIT Artificial Intelligence Laboratory
545 Technology Square
Cambridge, MA 02139, USA

Email: jcma@ai.mit.edu

References

1. RFC 1630 "Uniform Resource Identifiers in WWW", T. Berners-Lee, June 1994
2. RFC 1737 "Functional Requirements for Uniform Resource Names" K. Sollins, L. Masinter, December, 1994.
3. RFC 1738 "Uniform Resource Locators", T. Berners-Lee, L. Masinter, M. McCahill, December 1994
4. RFC 1945 "Hypertext Transfer Protocol -- HTTP/1.0", T. Berners-Lee, R. Fielding, H. Frystyk, May, 1996
5. RFC 2068 "Hypertext Transfer Protocol -- HTTP/1.1", R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, January 1997
6. RFC 2119 "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997
7. RFC 2168 "Resolution of Uniform Resource Identifiers using the Domain Name System", R. Daniel, M. Mealling, June 1997
8. RFC 2169 "A Trivial Convention for using HTTP in URN Resolution", R. Daniel, June 1997
9. RFC 2276 "Architectural Principles of Uniform Resource Name Resolution", K. Sollins, September, 1997
10. Internet Draft draft-ietf-http-ext-mandatory-00.txt "Mandatory Extensions in HTTP", H. Frystyk Nielsen, P. Leach, S. Lawrence, March 1998.