INTERNET-DRAFT
Date: March 13, 1998
Expires: September 18, 1998
draft-girod-urn-res-using-wire-00.txt

Lewis Girod
Benjie Chen
MIT Laboratory for Computer Science
John Mallery
MIT Artificial Intelligence Laboratory

URN Resolution Using WIRE

Status of this Memo

Abstract

The identifier resolution extensions proposed in WIRE [11] add new mechanisms for redirection and resolver delegation to HTTP. While these mechanisms are generalized to support resolution of all URIs, they were designed in part to meet the specific needs of URN resolution systems.

This document describes how the WIRE model maintains consistency with existing models of URN resolution, how WIRE can be used as a URN resolution protocol, and how WIRE fits into the existing URN resolution infrastructure, including NAPTR [8] and THTTP [9].

1 Introduction

This section comprises about half of this document. This lengthy introduction is an effort to summarize the history of URN resolution, explain the URN-oriented design rationale behind WIRE, and present a model for resolution that we have found useful for thinking about URN resolution while avoiding endless confusion.

The rest of the document is organized as follows: Section 2 describes the syntax and semantics of URN resolution hints; Section 3 describes some examples of URN resolution with WIRE; Section 4 describes the caching of URN resolution responses; Section 5 discusses security issues.

Details about URN syntax, URN resolution architecture, and particular URN resolution systems are outside the scope of this document. We expect readers of this document to be familiar with the various URN documents ([7][8][9][10]), HTTP ([4][5]), and WIRE ([11]).

1.1 Background

Uniform Resource Names, or URNs, are identifiers for Internet resources that have the capability to be maintained persistently for long periods of time. Each URN is globally unique and fits into a namespace that can be resolved globally. Rather than specifying a location and a fixed resolution mechanism, URNs are generally location-independent and are resolved using a mechanism that can evolve over time. Some URNs are considered to refer to "conceptual" resources that may have many different versions or may change dynamically. See [2] for a description of URN properties.

A large database containing resolution data for every URN is not a scalable approach to URN resolution. Instead, the URN Working Group of the IETF has been coordinating an effort to engineer a distributed authority model for URN namespaces. In this model, many resolvers worldwide serve different parts of a global URN namespace, implementing a federation of diverse resolution systems and protocols. This architecture enables the global URN infrastructure to scale to large numbers of URNs, to large volumes of requests, and to diverse administrative policies. This architecture requires that a URN resolver not only be able to resolve URNs within its own namespace, but also delegate spaces of URNs to other resolvers.

Until recently, URN resolution has exclusively been implemented using the Naming Authority PoinTeR (NAPTR) ([8]) resolution algorithm and the Trivial HTTP (THTTP) ([9]) convention for URI resolution. The NAPTR algorithm uses special records in the Domain Name System to achieve flexibility of delegation. There are two powerful advantages to this approach: first, the DNS is widely implemented and no significant changes to the server are required, and second, the DNS is designed with the desired mechanisms for distributing authority built in. Similarly, the THTTP convention is advantageous because it defines a simple interface for performing URN resolution that does can be added to an existing web server through the CGI interface.

However, early experimentation with the THTTP convention has shown some weaknesses. For example, it is not always clear how errors should be reported to the client, and it does not include a well-defined way to invoke methods other than "GET" on the referenced object. Redirection, which would be a powerful tool for delegation of URN spaces, is also not well defined by the convention. No recommendations are made about the behavior of caching proxies or client side caches.

WIRE [11] is an attempt to integrate functionality necessary for URN resolution into HTTP and to reduce that functionality to the more general case of URI resolution. WIRE is an extension of HTTP/1.0 or above that defines new semantics for resolver delegation. WIRE combines the functionality of a "terminal" resolution protocol such as THTTP with redirection and delegation functionality similar to that of the NAPTR algorithm. WIRE can both act as a gateway to other systems as well as return delegation responses that can "escape" to arbitrary alternate protocols.

1.2 Requirements

Work in the IETF URN Working Group has led to a set of requirements for URN resolution infrastructure [2][10]. WIRE was designed with the following goals in mind:

1. Enable new resolver implementations to work immediately with the existing infrastructure. WIRE must act as a gateway interface to new resolution services, while permitting new services to independently choose whether to proxy a resolution request or redirect it to another resolver.
2. Permit resolver delegations to reference other protocols or services. A resource may be accessible via protocols other than HTTP (e.g. CORBA objects). Some resolution protocols may specify local caching resolvers (e.g. HTTP clients can be configured to use a caching proxy; a web client might use the local DNS server to resolve NAPTR records).
3. Encourage the development of hint-based mechanisms by defining new data structures that bind hint information to URIs. Implement the idea of "delegation" as resolution to incremental metadata, and unify that idea with "terminal" resolution to resources. Specify mechanisms whereby client-side interpretation of metadata can act as a vehicle for delegation, redirection, and content negotiation.

URN infrastructure enables persistent references by building flexibility into URN resolution mechanisms. In order to speed deployment, these mechanisms need to be incorporated into existing infrastructures. The W3 infrastructure is a particularly appealing choice because of its wide acceptance and its potential to benefit from persistent naming. A number of reasons make adding URN resolution to HTTP a viable approach:

1. HTTP is a very widely deployed protocol. It is simple and human readable. It defines proxy semantics that can be used to implement new features in a proxy server. WIRE extends HTTP and therefore retains those properties.
2. HTTP has many useful features and semantics that WIRE also inherit. These include proxy servers, caching, authentication, etc.
3. Through the use of resolution hints, WIRE gives clients the power to choose methods of resolution. This allows a flexible and scalable architecture.

1.3 Resolution Model

One of the recurring problems experienced by the URI community has been a tendency to get trapped in the cavernous ratholes surrounding references to resources. WIRE [11] includes a section describing a simplified resolution model for URI resolution. Here we explain how this model applies to URN resolution. In doing so we hope to address some of areas that have historically led to confusion.

Rather than repeat the description in [11], we summarize: to resolve a URN, ask the right server to resolve it, supplying any appropriate arguments, and that server will perform the resolution. If that server is not the right server, it returns a redirection/delegation, or an error message:

```
                        _____________________
  GET                  |                     |          Standard
   URN[?argument]  --->|   Right Resolver?   | Yes --->   HTTP
  [hint: "hint"]       |_____________________|          Response
          ^                 | No         | Error
          |                 |            |
          |                 v            v
          +-------  350 Redirect    400 Bad Request
```

The resolver in the box can do as much work on the client's behalf as its local policy allows. Arguments specified in the request can cause particular methods to be performed on the resource when it is resolved; for example "resolution services" can be implemented this way. The 350 redirect can redirect resolution to other protocols, which may in turn provide similar delegation capabilities. Just as WIRE's resolution model borrows many ideas from the NAPTR algorithm [8], a similar model could easily be applied to other resolution protocols.

1.3.1 Popular Modes of Confusion

This model is designed to avoid certain areas of confusion that crop up with URN resolution. First, it clears up confusion about what a URN resolves to, by declaring that a URN resolves to typed data, with the type indicated dynamically by response headers. In this way, the WIRE model rejects the idea that a URN is constrained to resolve to a particular type (e.g. a "special" metadata type), instead providing identical semantics to any other HTTP request. This allows WIRE to be used more readily as a gateway protocol.

The notion of "resolution services" [10] tends to further blur the question of what a URN resolves to. By implementing a resolver that supports a set of resolution services, in essence this means that the URN resolves to an object to which resolution services may be applied as method calls on an object. This is not a problem as long as it remains clear that the object named by a URN is not the "resource" that might eventually be retrieved (say, by servicing an "I2R" request) but rather the server-side instance that provides the resolution services. Historically this distinction has been the source of no small amount of confusion.

The idea of "metadata" is another source of confusion. It is clear that the association of metadata with an object is something that we would like to work into our more advanced and more flexible concept of identifier resolution. URNs can be mapped to collections of metadata that in turn reference more concrete resources; this metadata can be used by the client to do content negotiation, or can be exploited by indexing agents. All of these ideas stem from the possibility that a URN identifies a "conceptual resource", a resource that might move or change over time but can retain the same mapping from name to concept as a result of a flexible URN resolution mechanism. However, "conceptual resources" are problematic because it rapidly becomes unclear whether the URN references the metadata that describes the resource, the whole abstract resource, or any given part of the resource. Which should a resolution operation return?

1.3.2 An Axiomatic Model of Resolution

To make sense of these problems, we propose a set of axioms that define a way to think about resolution processes. These axioms look at resolution processes as a scoped set of processes that can be observed at varying levels of detail. Often this leads to confusion when there is ambiguity about which layer of abstraction is in focus and which stage of the discovery process is in process. The critical idea here is that the same URN might resolve differently at different layers of abstraction and stages of discovery. In order to understand how a particular URN should resolve it is important to keep the relevant contextual information clearly in mind.

1. In general, outside of the context of a particular resolution mechanism, a URI resolves to bits.
2. Within the context of a particular resolution mechanism, there are intermediate resolution steps and a final resolution step. External to that mechanism, we can ignore the intermediate steps, and consider the final step to be resolution of the URI.
3. In general, the final resolution step is a method call on some remote object, which may be as simple as "GET". The resolution mechanism defines whether the results of any given operation are best considered to be metadata, or data, or any other type of thing.
4. Any particular resolution process can be considered to be an intermediate step inside a more complex resolution process.
5. Different resolution mechanisms may resolve the same URI to different data. Different resolution mechanisms may consider "resolution" complete at different stages of the process.

1.3.3 Some Examples

These ideas are best explained with an example. A user clicks on a link that contains the text of URN X. The browser performs a sequence of intermediate steps to resolve URN X, resulting in a displayed HTML page. From the user's point of view, URN X resolves to that page.

If we zoom in on the browser, we see three intermediate steps, and we see that at this level X really resolves to metadata:

1. URN X resolves to bits that the browser understands as metadata
2. The metadata tells the browser that the English version is accessible

   at URN Y
3. Since the browser knows that the user wants the English version, it
   resolves URN Y to an HTML page that is in turn displayed.

If we zoom in on step (1), we see three intermediate steps, and we see that locally X resolves to a 350 response that points to resolver R:

1. A "GET X" request is sent to a local WIRE resolver.
2. That resolver returns a 350 redirect pointing to resolver R.
3. A "GET X" request is sent to resolver R, and a 200 response containing metadata is returned.

1.4 Clarification

In previous URN resolution documents ([8][10]) the term "Resolver Discovery System", or RDS, has been used to describe the interactions of a set of servers designed solely to locate a resolver, which can then be used to resolve the URN. In our opinion, this distinction between RDS and resolver unnecessarily increases the complexity of the model. It also puts an unnecessary constraint on designers of URN resolution systems: for example, some systems might include servers that can act both as resolvers and resolver locators.

In this document, we use the term "resolver" to describe a server that can either resolve a URN or delegate the resolution authority to another resolver. This second behavior allows us to use the term "resolver" as a replacement for "RDS".

2 Resolution Hints

WIRE defines a general form for resolution hints, and leaves the definition of specific hint schemes to the resolution applications that use them. In this section we present a hint syntax designed to support URN resolvers.

2.1 Resolution Starting Point

Namespace identifiers, stored in a global registry ([10]), identify URN namespaces. A namespace can be further divided into subspaces. When a child subspace and its parent namespace are covered by different resolvers, the resolution process of a URN in the child subspace must be delegated from the parent namespace resolver to the subspace resolver.

If a resolver covers more than one subspace, then prior to resolving the request URN it must first determine to which subspace the URN belongs. Because the lexical ordering of a URN is not guaranteed to be uniform across delegation boundaries (see section 4 for an example), a URN's enclosing subspace cannot be identified syntactically, and can only be reliably discovered by performing the entire resolution process from the beginning.

This means that in order to make progress in the resolution, the hint passed to a URN resolver may need to contain a token that tells the resolver which URN space to begin searching. We call this token a resolution starting point. A resolution starting point has the following grammar:

```
resolution-starting-pt = urn
```

The only syntactical constraint on resolution starting point is that it must follow the URN syntax ([7]). Its syntax and semantics are defined by the resolver that receives and interprets it. A child resolver that needs to receive a specific starting point specifies this to be part of the hint information served by the resolver of the parent namespace.

2.2 Resolution Hints for Resolving URNs

A resolution hint tells a client how to continue resolving a specific URI. When a client uses a hint to contact a new WIRE resolver, a derivative of that hint is included in the request to that resolver. For the purposes of URN resolution, a hint needs to contain the location and protocol of a resolver, and it may optionally specify a resolution starting point to be interpreted by that resolver and a type model pertaining to that resolver. Resolution hints for URN resolution have the following grammar:

```
resolution-hint       = "res-hint:" url [starting-point] [type-model]
starting-point        = ";scope=" resolution-starting-pt
type-model            = ";type=" type-specifier * ( "+" type-specifier )
type-specifier        = urn
```

Operationally speaking, the URL specified in the resolution hint tells the client how to contact the resolver. The type model tells the client what types of object are stored at the target resolver or what services are provided there. This can allow the client to invoke a particular method on the object through the '?' construct, if the type model is one that the client understands. The client can indicate which type(s) it wants to use by modifying the type parameter in the hint passed along in the request. If it is included in the original hint, the scope parameter helps the target resolver perform the search; if present this should be passed along to the target resolver unchanged.

For example, the resolution hint

```
res-hint:http://thebe.lcs.mit.edu;scope=urn:cid:;type=urn:type:N2C+urn:type:N2L
```

states that the URN can be further resolved at thebe.lcs.mit.edu using the HTTP WIRE extensions. The server expects the starting point "urn:cid:", and that the object there can perform methods corresponding to both "N2C" and "N2L" service types.

The protocol scheme of the URL portion of the resolution hint is used to specify the protocol used by the target resolver. Native resolver protocols can enable great performance improvements through more effective caching strategies, smarter cache sharing, and local caching servers. For example, NAPTR is efficient because the client is configured to query a local DNS server that can perform transparent caching, as well as numerous other performance tricks. Resolution hints might refer to the NAPTR client algorithm or the THTTP terminal resolution protocol:

```
res-hint:naptr://cid.urn.net
res-hint:thttp://thebe.lcs.mit.edu;type=urn:type:N2C+urn:type:N2L
```

Clients that understand the NAPTR algorithm could use the first hint to start following NAPTR records beginning with "cid.urn.net". Clients that understand THTTP could use the second hint to connect to a resolver and request either the N2C or N2L services.

The scope attribute in the hint is optional. In many instances it is unnecessary, for example in hints for servers that serve only a single, clearly defined subspace. If a resolver requires a scope attribute but it is missing, the resolver may either return an error or attempt to discover the URN's subspace beginning with the root URN registry, possibly returning a 350 **code as a result.**

For various purposes such as caching and loop avoidance ([11]), it is often desirable to determine if two resolution hints are the same. WIRE defines generic hint equivalence as lexical equivalence after URI normalization. For clients that understand the "res-hint" scheme, an extended set of normalization rules can be applied. Two resolution hints are lexically equivalent if they are octet-by-octet equal after the following preprocessing:

1. normalize the case of the leading "res-hint:" token, the ";scope=" token, and the ";type=" token.
2. %-escape the url portion using octet encoding rules from [3]
3. %-escape the urn portions using rules from [7]

2.3 Applications of Resolution Hints

2.3.1 Resolver Behaviors

A resolution hint is said to be "local to a resolver" if and only if the url portion of the hint identifies that resolver. Otherwise, it is a remote resolution hint.

If a resolver receives a resolution request with a local resolution hint, it must try to resolve the URN. If a request contains a remote resolution hint, the resolver may proxy the resolution request to the remote server indicated, according to local policy. If proxying the request violates local policy, the resolver may return a 400 error code.

When resolution is successful, normal HTTP semantics take over. A 404 error message should ONLY be returned if the resolver is the authoritative resolver but the URN cannot be resolved. When a resolver cannot resolve a URN because the URN is under a different subspace, a 350 response or a non-404 error message should be returned to the client. Encapsulated in this 350 **responses are a set of URIs, each bound to zero or more resolution** hints. Note that 350 responses may only be returned to clients that indicate compliance with WIRE by including the OPTIONAL header in the request. A resolver that receives a request from a non-WIRE client may choose to proxy the remainder of the resolution process according to local policy. Otherwise, a 400 response code must be returned.

When a URN resolver needs to delegate the resolution process, it should use a 350 response code to return a set of alternate URIs and hints that would allow the continuation of the current URN. These set should include hints for resolving the current URN and/or alternate URIs and hints that resolve to the same resource as the URN, where equivalence is defined by the authoritative resolver. The client should be able to use any of the URI bindings to complete the resolution process.

2.3.2 Client Behaviors

When a client receives a 350 response from a URN resolver, it should attempt to continue the resolution process by parsing the list of URI-hint bindings, and choose a URI and a resolution hint to use in a follow up request. If the protocol specified in the resolution hint indicates WIRE, then the followup request should request the selected URI and should include the hint in the Resolution-Hint request header. The client may use the type field to indicate which type or service it is attempting to use. If a protocol other than WIRE is specified in the hint, the client must follow the specification of that protocol.

2.3.3 Proxy Behaviors

Proxy resolvers can forward resolution requests with remote resolution hints to the remote resolver specified in the hint. When a resolver running in proxy mode receives a resolution request with a remote resolution hint, it should construct a resolution request using the request URI and the value of the Resolution-Hint header. That is, the location of the resolver to forward the request to should be the location specified by the hint. Furthermore, the Resolution-Hint header should also be forwarded to the remote resolver.

If a proxy resolver receives a URN resolution request without a hint, and the resolver cannot find an appropriate hint in its cache, it may reject the request, or it may attempt to discover a resolution hint by starting the resolution process from the root NID registry.

3 Examples

In this section we describe a few scenarios of URN resolution. In all these scenarios, we will try to resolve the name urn:cid:9802032044@thebe.lcs.mit.edu.

3.1 Trivial Example - No Cache

A client (C) would like to resolve the URN by making the following resolution request to a local URN resolver (R). Note, C does not send the Optional header, therefore R must assume C is not WIRE compliant.

```
GET urn:cid:9802032044@thebe.lcs.mit.edu HTTP/1.0
```

The local URN resolver is not the authoritative resolver, and did not receive a hint. Therefore it attempts to discover an authoritative resolver, first using NAPTR. A series of NAPTR lookups is performed by the resolver. At some point, the resolver receives a "p" record from NAPTR, and the record contains the resolution hint "http://urn.mit.edu;scope=urn:cid:mit:" The resolver then makes the following connection to urn.mit.edu

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

urn.mit.edu returns back a 350 delegation response:

```
HTTP/1.0 350 Resolution Delegated
Resolver-Location: "";"res-hint:http://
thebe.lcs.mit.edu/;scope=urn:cid:mit.lcs.thebe:"
Expires: Fri, 13 Mar 1998 17:00:00 GMT
```

The resolver R parses the Resolver-Location header, and makes a subsequent request to thebe.lcs.mit.edu

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

It turns out that thebe.lcs.mit.edu is the authoritative resolver, and returns a 200 response.

```
HTTP/1.0 200 OK

<entity>
```

The resolver R returns this response to the original client C.

3.2 Effects of Caching

Now that a request for that particular URN has been fulfilled, it is in the cache of the resolver R. Let's assume that the resolver R does not cache the final resolved entity, but it does remember the last 350 delegation and its expiration time. The next time a request from C to R for the same URN can be fulfilled by directly applying the 350 delegation response from cache. That is, the resolver R can directly contact the authoritative server at thebe.lcs.mit.edu, skipping the NAPTR lookups and the first WIRE lookup. Of course, any resolvers in the resolution sequence in section 3.1 can maintain their own caches.

3.3 Variation on a protocol

The resolver at thebe.lcs.mit.edu decides that HTTP is not secure enough, and it would like to use POP+Kerberos to perform resolution on its part of the cid namespace. In this case, the resolver at urn.mit.edu would have responded with

```
HTTP/1.0 350 Resolution Delegated
Resolver-Location: "";"res-hint:pop://
thebe.lcs.mit.edu/;scope=urn:cid:mit.lcs.thebe:;\
                   auth-method=kerberos/5"
Expires: Fri, 13 Mar 1998 17:00:00 GMT
```

If resolver R understands both POP and can perform KERBEROS authentication for the client, the resolver may perform a pop request and return the client C the resolved data. If the local resolver R does not understand POP, or if it cannot perform kerberos authentication for the client, it would have to return a 400 response to the client, and with an entity explaining the situation.

This example demonstrates that WIRE allows flexible switching of resolution protocols.

3.4 How WIRE interacts with NAPTR

Most of existing URN resolvers use the NAPTR ([8]) protocol. To reduce complexity, we assume any WIRE compliant client that wishes to resolve URNs also understands NAPTR. That is, it is up to the client to perform NAPTR lookups when necessary, and keep any internal state information necessary for NAPTR lookups.

A NAPTR record can point to a WIRE resolver by returning a record that looks like

```
;;      order pref flags service  regexp replacement
 IN NAPTR 100  10  "p" "WIRE+N2R" ""     http://urn.mit.edu;scope=urn:cid:mit:
```

To switching back to NAPTR, we can use a resolution hint that looks like

```
res-hint:NAPTR://z3950.gatech.edu
```

4 Caching

URN resolution using WIRE has a great need for caching 350 responses because of the number of delegations each URN resolution process may involve. URN resolution using WIRE should follow the same cache semantics as defined for WIRE ([11]).

It is an unfortunate consequence of the flexibility of URN delegation that it is presently impossible to specify a generalized form for wildcards that can describe delegated subsets of URN space. URN delegation generally does not occur along a single model of hierarchy, for a variety of reasons including the need to grandfather existing persistent namespaces and the need to permit delegation structures to evolve and fragment over time.

For example, within a single namespace different portions of the syntax can exhibit different directions of hierarchy. In the following diagram, consider one possible grandfathered namespace incorporating content-IDs. (The notation L.R means left-to-right hierarchy delimited by '.', while R@L means right-to-left hierarchy delimited by '@'.)

```
        urn:cid:9802030121.AA21353@thebe.lcs.mit.edu
        <-L:R--------------------------------------->
                <-R@L------------------------------->
                <-L.R------------> <-R.L----------->
```

This means that, with the exception of circumstances where the cache agent understands the delegation structures involved, it is not possible to cache information about spaces of URNs.

5 Security Considerations

WIRE security considerations are discussed in [11].

6 Acknowledgments

The requirements for URN resolution that this protocol intends to address have been discussed and developed in meetings of the IETF URN Working Group, as well as in separate conversations among its many members. The emphasis on extensibility and delegation via metadata in 350 redirects is an implementation of RDS switching ideas that are described in the URN architecture document, RFC 2276 [10]. This in turn has its roots in the "Knoxville Compromise" which defined a basic architecture for URN resolution.

A great deal of the design of this protocol and the accompanying participant behaviors is inherited directly from the NAPTR algorithm and record format. The design of the metadata response format includes similar notions of preference and order; the service request header copies the syntax for specifying services in NAPTR. Following the example of NAPTR, this protocol treats URN resolution as a discovery process, emphasizing that throughout the discovery process some portion of a URN is opaque and does not have a strict syntax imposed from above.

Henrik Frystk Nielson provided invaluable feedback and comments on an original version of this document. Discuss with Henrik on the concept of resolution and the generic URI resolution model led to the current versions of this document and the WIRE specification.

Last but not the least, Karen Sollins and Dorothy Curtis have provided many insightful ideas and feedback on the general URN resolution architecture and on this document.

7. Authors Addresses

Lewis Girod
Benjie Chen

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

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 2141 "URN Syntax", R. Moats. May 1997
8. RFC 2168 "Resolution of Uniform Resource Identifiers using the Domain Name System", R. Daniel, M. Mealling, June 1997
9. RFC 2169 "A Trivial Convention for using HTTP in URN Resolution", R. Daniel, June 1997
10. RFC 2276 "Architectural Principles of Uniform Resource Name Resolution", K. Sollins, September, 1997
11. Internet Draft draft-girod-w3-id-res-ext-00 "WIRE: W3 Identifier Resolution Extensions", L. Girod, B. Chen, H. Frystyk, J. Mallery, March 1998 (work in progress)
12. Internet Draft draft-ietf-urn-resolution-services-05 "URI Resolution Services Necessary for URN Resolution" M. Mealling, March, 1998 (work in progress)