

Network Working Group
Request for Comments: 2169
Category: Experimental

R. Daniel
Los Alamos National Laboratory
June 1997

A Trivial Convention for using HTTP in URN Resolution

Status of this Memo
=====

This memo defines an Experimental Protocol for the Internet community. This memo does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Abstract:
=====

The Uniform Resource Names Working Group (URN-WG) was formed to specify persistent, location-independent names for network accessible resources, as well as resolution mechanisms to retrieve the resources given such a name. At this time the URN-WG is considering one particular resolution mechanism, the NAPTR proposal [1]. That proposal specifies how a client may find a "resolver" for a URN. A resolver is a database that can provide information about the resource identified by a URN, such as the resource's location, a bibliographic description, or even the resource itself. The protocol used for the client to communicate with the resolver is not specified in the NAPTR proposal. Instead, the NAPTR resource record provides a field that indicates the "resolution protocol" and "resolution service requests" offered by the resolver.

This document specifies the "THTTP" resolution protocol - a trivial convention for encoding resolution service requests and responses as HTTP 1.0 or 1.1 requests and responses. The primary goal of THTTP is to be simple to implement so that existing HTTP servers may easily add support for URN resolution. We expect that the databases used by early resolvers will be useful when more sophisticated resolution protocols are developed later.

1.0 Introduction:
=====

The NAPTR specification[1] defined a new DNS resource record which may be used to discover resolvers for Uniform Resource Identifiers. That resource record provides the "services" field to specify the "resolution protocol" spoken by the resolver, as well as the "resolution services" it offers. Resolution protocols mentioned in

that specification are Z3950, THTTP, RCDS, HDL, and RWHOIS. (That list is expected to grow over time). The NAPTR specification also lists a variety of resolution services, such as N2L (given a URN, return a URL); N2R (Given a URN, return the named resource), etc.

This document specifies the "THTTP" (Trivial HTTP) resolution protocol. THTTP is a simple convention for encoding resolution service requests and responses as HTTP 1.0 or 1.1 requests and responses. The primary goal of THTTP is to have a URN resolution protocol that can easily be added to existing HTTP daemons. Other resolution protocols are expected to arise over time, so this document serves a secondary purpose of illustrating the information that needs to be specified for a URN resolution protocol. One of the resolution protocols we expect to be developed is an extension of HTTP with new methods for the resolution services. Therefore, we use "THTTP" as the identifier for this protocol to leave "HTTP" for later developments.

The reader is assumed to be familiar with the HTTP/1.0 [2] and 1.1 [3] specifications. Implementors of this specification should be familiar with CGI scripts, or server-specific interfaces, for database lookups.

2.0 General Approach:

=====

The general approach used to encode resolution service requests in THTTP is quite simple:

```
GET /uri-res/<service>?<uri> HTTP/1.0
```

For example, if we have the URN "urn:foo:12345-54321" and want a URL, we would send the request:

```
GET /uri-res/N2L?urn:foo:12345-54321 HTTP/1.0
```

The request could also be encoded as an HTTP 1.1 request. This would look like:

```
GET /uri-res/N2L?urn:foo:12345-54321 HTTP/1.1
Host: <whatever host we are sending the request to>
```

Responses from the HTTP server follow standard HTTP practice. Status codes, such as 200 (OK) or 404 (Not Found) shall be returned. The normal rules for determining cachability, negotiating formats, etc. apply.

Handling these requests on the server side is easy to implement using CGI or other, server-specific, extension mechanisms. CGI scripts will see the incoming URI in the QUERY_STRING environment variable. Any %encoded characters in the URN will remain in their %encoded state in that string. The script can take the URN, look it up in a database, and return the requested information.

One caveat should be kept in mind. The URN syntax document [4] discusses the notion of lexical equivalence and requires that resolvers return identical results for URNs that are lexically equivalent. Implementors of this specification must be careful to obey that rule. For example, the two requests below MUST return identical results, since the URNs are lexically equivalent.

```
GET /uri-res/N2L?urn:cid:foo@huh.com HTTP/1.0
```

```
GET /uri-res/N2L?URN:CID:foo@huh.com HTTP/1.0
```

3.0 Service-specific details:

=====

This section goes through the various resolution services established in the URN services document [5] and states how to encode each of them, how the results should be returned, and any special status codes that are likely to arise.

Unless stated otherwise, the THTTP requests are formed according to the simple convention above, either for HTTP/1.0 or HTTP/1.1. The response is assumed to be an entity with normal headers and body unless stated otherwise. (N2L is the only request that need not return a body).

3.1 N2L (URN to URL):

The request is encoded as above. The URL MUST be returned in a Location: header for the convenience of the user in the most common case of wanting the resource. If the lookup is successful, a 30X status line SHOULD be returned. HTTP/1.1 clients should be sent the 303 status code. HTTP/1.0 clients should be sent the 302 (Moved temporarily) status code unless the resolver has particular reasons for using 301 (moved permanently) or 304 (not modified) codes.

Note that access controls may be applied to this, or any other, resolution service request. Therefore the 401 (unauthorized) and 403 (forbidden) status codes are legal responses. The server may wish to provide a body in the response to explain the reason for refusing access, and/or to provide alternate information about the resource, such as the price it will cost to obtain the resource's URL.

3.2 N2Ls (URN to URLs):

The request is encoded as above. The result is a list of 0 or more URLs. The Internet Media Type (aka ContentType) of the result may be negotiated using standard HTTP mechanisms if desired. At a minimum the resolver should support the text/uri-list media type. (See Appendix A for the definition of this media type). That media type is suitable for machine-processing of the list of URLs. Resolvers may also return the results as text/html, text/plain, or any other media type they deem suitable.

No matter what the particular media type, the result MUST be a list of the URLs which may be used to obtain an instance of the resource identified by the URN. All URIs shall be encoded according to the URI specification [6].

If the client has requested the result be returned as text/html or application/html, the result should be a valid HTML document containing the fragment:

```
<UL>
<LI><A HREF="...url 1...">...url 1...</A>
<LI><A HREF="...url 2...">...url 2...</A>
etc.
</UL>
```

where the strings ...url n... are replaced by the n'th URL in the list.

3.3 N2R (URN to Resource):

The request is encoded as above. The resource is returned using standard HTTP mechanisms. The request may be modified using the Accept: header as in normal HTTP to specify that the result be given in a preferred Internet Media Type.

3.4 N2Rs (URN to Resources):

This resolution service returns multiple instances of a resource, for example, GIF and JPEG versions of an image. The judgment about the resources being "the same" resides with the naming authority that issued the URN.

The request is encoded as above. The result shall be a MIME multipart/alternative message with the alternative versions of the resource in separate body parts. If there is only one version of the resource identified by the URN, it MAY be returned without the

multipart/alternative wrapper. Resolver software SHOULD look at the Accept: header, if any, and only return versions of the resource that are acceptable according to that header.

3.5 N2C (URN to URC):

URCs (Uniform Resource Characteristics) are descriptions of other resources. This request allows us to obtain a description of the resource identified by a URN, as opposed to the resource itself. The description might be a bibliographic citation, a digital signature, a revision history, etc. This document does not specify the content of any response to a URC request. That content is expected to vary from one resolver to another.

The format of any response to a N2C request MUST be communicated using the ContentType header, as is standard HTTP practice. The Accept: header SHOULD be honored.

3.6 N2Ns (URN to URNs):

While URNs are supposed to identify one and only one resource, that does not mean that a resource may have one and only one URN. For example, consider a resource that has something like "current-weather-map" for one URN and "weather-map-for-datetime-x" for another URN. The N2Ns service request lets us obtain lists of URNs that are believed equivalent at the time of the request. As the weathermap example shows, some of the equivalences will be transitory, so the standard HTTP mechanisms for communicating cachability MUST be honored.

The request is encoded as above. The result is a list of all the URNs, known to the resolver, which identify the same resource as the input URN. The result shall be encoded as for the N2Ls request above (text/uri-list unless specified otherwise by an Accept: header).

3.7 L2Ns (URL to URNs):

The request is encoded as above. The response is a list of any URNs known to be assigned to the resource at the given URL. The result shall be encoded as for the N2Ls and N2Ns requests.

3.8 L2Ls (URL to URLs):

The request is encoded as described above. The result is a list of all the URLs that the resolver knows are associated with the resource located by the given URL. This is encoded as for the N2Ls, N2Ns, and L2Ns requests.

3.9 L2C (URL to URC):

The request is encoded as above, the response is the same as for the N2C request.

Appendix A: The text/uri-list Internet Media Type

=====

[This appendix will be augmented or replaced by the registration of the text/uri-list IMT once that registration has been performed].

Several of the resolution service requests, such as N2Ls, N2Ns, L2Ns, L2Ls, result in a list of URIs being returned to the client. The text/uri-list Internet Media Type is defined to provide a simple format for the automatic processing of such lists of URIs.

The format of text/uri-list resources is:

- 1) Any lines beginning with the '#' character are comment lines and are ignored during processing. (Note that '#' is a character that may appear in URIs, so it only denotes a comment when it is the first character on a line).
- 2) The remaining non-comment lines MUST be URIs (URNs or URLs), encoded according to the URI specification RFC[6]. Each URI shall appear on one and only one line.
- 3) As for all text/* formats, lines are terminated with a CR LF pair, although clients should be liberal in accepting lines with only one of those characters.

In applications where one URI has been mapped to a list of URIs, such as in response to the N2Ls request, the first line of the text/uri-list response SHOULD be a comment giving the original URI.

An example of such a result for the N2L request is shown below in figure 1.

```
# urn:cid:foo@huh.org
http://www.huh.org/cid/foo.html
http://www.huh.org/cid/foo.pdf
ftp://ftp.foo.org/cid/foo.txt
```

Figure 1: Example of the text/uri-list format

Appendix B: n2l.pl script

=====

This is a simple CGI script for the N2L resolution service. It assumes the presence of a DBM database to store the URN to URL mappings. This script does not specify standard behavior, it is provided merely as a courtesy for implementors. In fact, this script does not process incoming Accept: headers, nor does it generate status codes. Such behavior should be part of a real script for any of the resolution services.

```
#!/bin/perl
# N2L - performs urn to url resolution

$n2l_File = "...filename for DBM database...";

$urn = $ENV{'QUERY_STRING'} ;

# Sanity check on the URN. Minimum length of a valid URN is
# 7 characters - "urn:", a 1-character Namespace ID, ":", and
# a 1-character namespace-specific string. More elaborate
# sanity checks should be part of a real resolver script.
if(length($urn)<7)
{
    $error=1;
}

if(!$error)
{
    # Convert lexically equivalent versions of a URI into
    # a canonical version for DB lookups.
    $urn =~ s/^urn:([^\:]*):(.*)$/sprintf("urn:%s:%s", lc $1, $2)/ie;

    dbmopen(%lu,$n2l_File,0444);
    if($lu{$urn})
    {
        $url=$lu{$urn};
        print STDOUT "Location: $url\n\n";
    }else{
        $error=2;
    }
    dbmclose(%lu);
}

if($error)
{
    print "Content-Type: text/html \n\n";
    print "<html>\n";
    print "<head><title>URN Resolution: N2L</title></head>\n";
    print "<BODY>\n";
    print "<h1>URN to URL resolution failed for the URN:</h1>\n";
    print "<hr><h3>$urn</h3>\n";
    print "</body>\n";
    print "</html>\n";
}

exit;
```


References:

=====

- [1] Daniel, Ron and Michael Mealling, RFC 2168, "Resolution of Uniform Resource Identifiers using the Domain Name System", June 1997.
- [2] Berners-Lee, T, R. Fielding, H. Frystyk, RFC 1945, "Hypertext Transfer Protocol -- HTTP/1.0", T. Berners-Lee, May 1996.
- [3] Fielding, R., J. Gettys, J.C. Mogul, H. Frystyk, T. Berners-Lee, RFC 2068, "Hypertext Transfer Protocol -- HTTP/1.1", Jan. 1997.
- [4] Moats, R., RFC 2141, "URN Syntax", May 1997.
- [5] URN-WG. "URN Resolution Services". Work In Progress.
- [6] Berners-Lee, T., RFC 1630, "Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web", June 1994.

Security Considerations

=====

Communications with a resolver may be of a sensitive nature. Some resolvers will hold information that should only be released to authorized users. The results from resolvers may be the target of spoofing, especially once electronic commerce transactions are common and there is money to be made by directing users to pirate repositories rather than repositories which pay royalties to rightsholders. Resolution requests may be of interest to traffic analysts. The requests may also be subject to spoofing.

The requests and responses in this draft are amenable to encoding, signing, and authentication in the manner of any other HTTP traffic.

Author Contact Information:

=====

Advanced Computing Lab, MS B287
Los Alamos National Laboratory
Los Alamos, NM, USA, 87545
voice: +1 505 665 0597
fax: +1 505 665 4939
email: rdaniel@lanl.gov

