Network Working Group
Request for Comments: 2483
Category: Experimental

M. Mealling
Network Solutions, Inc.
R. Daniel, Jr.
Los Alamos National Laboratory

January 1999

URI Resolution Services Necessary for URN Resolution

Status of this Memo

Copyright Notice

Abstract

Retrieving the resource identified by a Uniform Resource Identifier (URI) [1] is only one of the operations that can be performed on a URI. One might also ask for and get a list of other identifiers that are aliases for the original URI or a bibliographic description of the resource the URI denotes, for example. This applies to both Uniform Resource Names (URNs) and Uniform Resource Locators (URLs). Uniform Resource Characteristics (URCs) are discussed in this document but only as descriptions of resources rather than identifiers.

A service in the network providing access to a resource may provide one or some of these options, but it need not provide all of them. This memo specifies an initial set of these operations that can be used to describe the interactions provided by a given access service. It also suggests guidelines that should be adhered to when those operations are encoded in a protocol.

1. Introduction

In the course of formulating current proposals [2] regarding URNs [3], it became apparent that requiring servers to manage all of the desired functions or requiring clients to process varied information returned by a server was unrealistic and a barrier to adoption. There needed to be some way for a client to be able to identify a server that specialized in the complex and another that specialized in the

simple (but fast). Also, in subsequent conversations it became obvious that, in most cases, some of the operations were inappropriate or difficult for certain identifiers.

The Problem

In the process of learning about a resource in the Internet, there are a variety of possible functions that may be important and/or useful, such as discovery of locators, names, descriptions, and accessing the resource itself. A given service may support only a subset of these; hence, it is important to describe such an access service by the types of functions supported and the resources of which it has some knowledge. For example, in the framework for an RDS described in [5] the RDS itself may provide URLs [6][7], while the resolvers may provide descriptions, URLs, or even the resources themselves. The design of an RDS, as proposed in RFC 2168 [2], may be more generous and provide all of the above.

This problem requires some well understood set of identifiers that specify those operations. But an exhaustive set would both be impossible and not very necessary. Thus, this document will list several operations, as well as, lay out requirements for specifying new operations.

The purpose of this document is to define a list of such functions and short names for them and then use them in defining the interface to an access service. Previous versions of this document referred to services where the arguments were specific types of URIs such as URNs or URLs. These services were called "N2L" and "L2L",for example. Their use has been changed in favor of the more general URI form.

Design Criteria

To meet these requirements a fairly simple design criteria was used. The need to identify the operation with some token such that its operands, algorithm, and errors were known proved sufficient to meet these requirements.

2. General Specification

To provide a framework both for the specifications in this document and for future work to be written by others, the guidelines below are suggested for documents that seek to specify new operations. Any specification of a member of this set of operations should address these issues with respect to its operands, algorithm, output, and errors.

Due to the small number of listed functions, a registration mechanism was dismissed as premature. If this list grows, a registration mechanism will probably be needed.

Also, due to the experimental nature of this document and the systems that use its specifications, the use of words like MUST and SHALL are limited. Where used they reflect a case where this specification could cause harm to existing, non-experimental systems such as HTTP and URNs. Thus, 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.

2.1 Operands

Operands must contain the following pieces of information:

* name of the operation
* case insensitive mnemonic for the operation
* number of operands
* type of each operand
* format of each operand

2.2 Algorithm

The exact algorithm for the operation must either be specified completely or it must be considered opaque and defined by the server or application.

2.3 Output

Output must specify one of the following:

* there is no output
* the output is undefined
* the output itself and its content
* the fact that the output is an object and the object’s type and format
* any non-protocol specific errors

2.4 Error Conditions

All errors that are considered applicable across all implementations and application environments must be included. Errors that depend on the system conveying the service are not included. Thus, many of the expected errors such as service availability or operation syntax are not included in this document since they are implementation dependent.

2.5 Security Considerations

Any security considerations relating to the service provided must be specified. This does NOT include considerations dealing with the protocol used to convey the service or to those that normally accompany the results of the service. For example, a service that returned a single URL would need to discuss the situation where someone maliciously inserts an incorrect URL into the resolver but NOT the case where someone sends personal information across the Internet to the resource identified by the correct URL.

3. Encoding The Operations

To be useful, these operations have to be used within some system or protocol. In many cases, these systems and protocols will place restrictions on which operations make sense and how those that do are syntactically represented. It is sufficient for those protocols to define new operations within their own protocol specification documents but care should be taken to make this fact well known.

Also, a given system or protocol will have its own output specifications that may restrict the output formats of a given operation. Additionally, a given protocol may have better solution for output than the ones given here. For example, the result of an operation that converts a URI to more than one URL may be encoded in a protocol-specific manner that conveys information about the closeness of each resource on the network.

Thus, the requirements on encoding these operations within a given system are as follows:

* which subset of the operations are allowed
* how the operator is encoded
* how the operands are encoded
* how the error codes are returned

The text/uri-list MIME Media Type is specified in Section 5. This Media Type is merely a suggestion for experimental systems that need a simple implementation. It is included here merely as an example to show completeness (however simple it may be).

4. The Incomplete Set

4.1 I2L (URI to URL)

* Name: URI to URL
* Mnemonic: I2L
* Number of Operands: 1
* Type of Each Operand: First operand is a URI.
* Format of Each Operand: First operand is encoded as a URI.
* Algorithm: Opaque
* Output: One and only one URL
* Errors Conditions:
    - Malformed URI
    - URI is syntactically valid but does not exist in any form.
    - URI exists but there is no available output from this operation.
    - URI existed in the past but nothing is currently known about it.
    - Access denied

* Security Considerations:
    - Malicious Redirection
      One of the fundamental dangers related to any service such as this is that a malicious entry in a resolver's database will cause clients to resolve the URI into the wrong URL. The possible intent may be to cause the client to retrieve a resource containing fraudulent or damaging material.
    - Denial of Service
      By removing the URL to which the URI maps, a malicious intruder may remove the client's ability to retrieve the resource.

This operation is used to map a single URI to a single URL. It is used by lightweight clients that do not have the ability to select from a list of URLs or understand a URC. The algorithm for this mapping is dependent on the URI scheme.

4.2 I2Ls (URI to URLs)

* Name: URI to URLs
* Mnemonic: I2LS
* Number of Operands: 1
* Type of Each Operand: First operand is a URI.
* Format of Each Operand: First operand is encoded as a URI.
* Algorithm: Opaque
* Output: A list of zero or more URLs
* Errors:
    - Malformed URI

        o URI is syntactically valid but does not exist in any form.
        o URI exists but there is no available output from this operation.
        o URI existed in the past but nothing is currently known about it.
        o Access denied
    * Security Considerations:
        o Malicious Redirection (see I2L)
        o Denial of Service (see I2L)

This operation is used to map a single URI to 0 or more URLs. It is used by a client that can pick from a list of URLs based on some criteria that are important to the client. The client should not make any assumptions about the order of the URLs returned. No matter what the particular media type, the result should be a list of the URLs that may be used to obtain an instance of the resource identified by the URI. All URIs shall be encoded according to the URL [7] and URN [3] specifications.

4.3 I2R (URI to Resource)

* Name: URI to Resource
* Mnemonic: I2R
* Number of Operands: 1
* Type of Each Operand: First operand is a URI.
* Format of Each Operand: First operand is encoded as a URI.
* Algorithm: Opaque
* Output: An instance of the resource named by the URI.
* Errors:
    o Malformed URI
    o URI is syntactically valid but does not exist in any form.
    o URI exists but there is no available output from this operation.
    o URI existed in the past but nothing is currently known about it.
    o Access denied
* Security Considerations:
    o Malicious Redirection (see I2L)
    o Denial of Service (see I2L)

This operation is used to return a single instance of the resource that is named by the URI. The format of the output is dependent on the resource itself.

4.4 I2Rs (URI to Resources)

* Name: URI to Resources
* Mnemonic: I2Rs
* Number of Operands: 1
* Type of Each Operand: First operand is a URI.
* Format of Each Operand: First operand is encoded as a URI.
* Algorithm: Opaque
* Output: Zero or more instances of the resource named by the URI.
* Errors:
    - o Malformed URI
    - o URI is syntactically valid but does not exist in any form.
    - o URI exists but there is no available output from this operation.
    - o URI existed in the past but nothing is currently known about it.
    - o Access denied
* Security Considerations:
    - o Malicious Redirection (see I2L)
    - o Denial of Service (see I2L)

This operation is used to return 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 URI.

The output shall be a MIME multipart/alternative [4] 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.

4.5 I2C (URI to URC)

* Name: URI to URC * Mnemonic: I2C * Number of Operands: 1 * Type of Each Operand: First operand is a URI. * Format of Each Operand: First operand is encoded as a URI. * Algorithm: Opaque * Output: A URC * Errors:
    - o Malformed URI
    - o URI is syntactically valid but does not exist in any form.
    - o URI exists but there is no available output from this operation.
    - o URI existed in the past but nothing is currently known about it.
    - o Access denied * Security Considerations:
    - o Malicious Redirection (see I2L)
    - o Denial of Service (see I2L)

Uniform Resource Characteristics are descriptions of resources. This request allows the client to obtain a description of the resource identified by a URI, as opposed to the resource itself or simply the resource's URLs. The description might be a bibliographic citation, a digital signature, or a revision history. This memo does not specify the content of any response to a URC request. That content is expected to vary from one server to another.

4.6 I2CS (URI to URCs)

* Name: URI to URCs
* Mnemonic: I2CS
* Number of Operands: 1
* Type of Each Operand: First operand is a URI.
* Format of Each Operand: First operand is encoded as a URI.
* Algorithm: Opaque
* Output: Zero or more URCs
* Errors:
    - Malformed URI
    - URI is syntactically valid but does not exist in any form.
    - URI exists but there is no available output from this operation.
    - URI existed in the past but nothing is currently known about it.
    - Access denied
* Security Considerations:
    - Malicious Redirection (see I2L)
    - Denial of Service (see I2L)

URCs can come in different formats and types. This operation returns zero or more URCs that are appropriate for the given URI.

4.7 I2N (URI to URN)

* Name: URI to URN
* Mnemonic: I2N
* Number of Operands: 1
* Type of Each Operand: First operand is a URN.
* Format of Each Operand: First operand is encoded as a URI.
* Algorithm: Opaque
* Output: One and only one URN
* Errors:
    - Malformed URI
    - URI is syntactically valid but does not exist in any form.
    - URI exists but there is no available output from this operation.
    - URI existed in the past but nothing is currently known about it.

        - Access denied
    * Security Considerations:
        - Malicious Redirection (see I2L)
        - Denial of Service (see I2L)

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 one organization wishes to name 'foo'; another organization, in agreement with the first, wants to call the resource 'bar'. Both organizations can agree that both names 'name' the same resource and that the URNs 'foo' and 'bar' are equivalent.

The result is a URN, known to the server, that identifies the same resource as the input URN.

Extreme care should be taken with this service as it toys with the idea of equality with respect to URNs. As mentioned in several URN documents, the idea of equality is very domain specific. For example, a URN pointing to a weather map for a particular day and a URN pointing to the map as it changes from day to day would NOT be returned in this example because they point to do different resources. Some other concept of temporary equivalence is at work. This service instead deals with resources that have two different names where there is a binding between the names that is agreed by both name assigners. I.e., both namespaces MUST have agreed that the each name can be used in place of the other and the meaning does not change.

4.8 I2Ns (URI to URNs)

    * Name: URI to URNs
    * Mnemonic: I2NS
    * Number of Operands: 1
    * Type of Each Operand: First operand is a URI.
    * Format of Each Operand: First operand is encoded as a URI.
    * Algorithm: Opaque
    * Output: A list of URNs
    * Errors:
        - Malformed URI
        - URI is syntactically valid but does not exist in any form.
        - URI exists but there is no available output from this operation.
        - URI existed in the past but nothing is currently known about it.
        - Access denied
    * Security Considerations:
        - Malicious Redirection (see I2L)

        o Denial of Service (see I2L)

This operation simply returns zero or more URNs following the same criteria and cautions as the I2N operation.

4.9 I=I (Is URI equal to URI):

* Name: URI = URI
* Mnemonic: I=I
* Number of Operands: 2
* Type of Each Operand: Both operands are URIs.
* Format of Each Operand: Both operands are encoded as a URIs.
* Algorithm: Opaque
* Output: TRUE or FALSE
* Errors:
    - Malformed URIs
    - URIs are syntactically valid but do not exist in any form.
    - URIs exist but there is no available output from this operation.
    - URIs existed in the past but nothing is currently known about them.
    - Access denied
* Security Considerations:
    - Malicious Redirection (see I2L)
    - Denial of Service (see I2L)

This operation is used to determine whether two given URIs are considered to be equal by the server being asked the question. The algorithm used to determine equality is opaque. No assertions are made about whether or not the URIs exhibits characteristics of URNs or URLs.

5. The text/uri-list Internet Media Type

Several of the resolution service requests, such as I2Ls, I2Ns, 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.

This is a copy of the IANA registration of the text/uri-list Media Type.

Date: Fri, 18 Apr 97 08:36:07 PDT
From: Ron Daniel Jr. <rdaniel@lanl.gov>
To: iana@iana.org, rdaniel@lanl.gov
Subject: Request for MIME media type Text/IETF Tree - uri-list

Name : Ron Daniel Jr.

E-mail : rdaniel@lanl.gov

MIME media type name : Text

MIME subtype name : IETF Tree -uri-list

Required parameters : none

Optional parameters : charset

Currently, URIs can be represented using US-ASCII. However, there are many non-standard URIs which use special character sets. Discussion of how to best achieve internationalization of URIs is underway. This registration will be updated with a discussion of the URI charsets once that discussion has concluded.

Encoding considerations : Some transfer protocols, such as SMTP, place limits on the length of lines. Very long URIs might exceed those limits. Systems must therefore be prepared to use a suitable content transfer encoding. This is anticipated to be a rare occurance.

Security considerations : Client software should be aware of the security considerations of URIs. For example, accessing some URIs can result in sending a death threat to a head of state, frequently prompting a visit from the relevant protective service. Accessing other URIs may result in financial obligations, or access to resources considered inappropriate by one’s employer.

While the legitimate provider of a uri-list could exploit these properties for good or ill, it is more likely that uri-lists will be falsified in order to exploit such characteristics of URIs.

Additionally, the lookup and reverse lookup potential of the uri-list may be attractive to traffic analysts. URI lists may also reveal confidential information, such as the location of sensitive information.

Because of these considerations, external confidentiality measures should be available to protect uri-list responses when appropriate.

Interoperability considerations : none known

Published specification : Uniform Resource Locators (URLs) and Uniform Resource Names (URNs) are two instances of the more general class of identifiers known as Uniform Resource Identifiers (URIs). URN resolution methods frequently wish to return lists of URLs for a resource so that fault-tolerance and load balancing can be achieved. The text/uri-list format is intended to be a very simple format for communicating such lists of URLs (and URNs) in a form suitable for automatic processing.

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 URIs may contain the '#' character, so it is only a comment character when it is the first character on a line.)

2) The remaining non-comment lines shall be URIs (URNs or URLs), encoded according to the URL or URN specifications (RFC2141, RFC1738 and RFC2396). Each URI shall appear on one and only one line. Very long URIs are not broken in the text/uri-list format. Content-transfer-encodings may be used to enforce line length limitations.

3) As for all text/* formats, lines are terminated with a CRLF pair.

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

An example of the format is given below:

```
  # urn:isbn:0-201-08372-8
  http://www.huh.org/books/foo.html
  http://www.huh.org/books/foo.pdf
  ftp://ftp.foo.org/books/foo.txt
```

Applications which use this media : URN resolvers are the initial applications. Web clients and proxies are applications that are likely to support this format in the future.

Additional information :

1. Magic number(s) : none at this time
2. File extension(s) : .uris or .uri recommended
3. Macintosh file type code : URIs recommended

This media type is the product of the URN working group of the IETF.

Person to contact for further information :

1. Name : Ron Daniel Jr.
2. E-mail : rdaniel@lanl.gov

Intended usage : Limited Use
The text/uri-list media type is intended for use in applications which utilize URIs for replicated resources.

Author/Change controller : Ron Daniel Jr.
Los Alamos National Laboratory
rdaniel@lanl.gov

In applications where one URI has been mapped to a list of URIs, such as in response to the I2Ls 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 I2L request is shown below in Figure 1.

6. Security Considerations

Communications with a server may be of a sensitive nature. Some servers will hold information that should only be released to authorized users. The results from servers 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 that pay royalties to rights-holders. Server requests may be of interest to traffic analysts. The requests may also be subject to spoofing.

The "Access denied" error message assumes a system within which the operation is being performed that can convey an authenticated concept of access control. Thus, the "Access denied" message should only be returned by systems that have an appropriate method of determining access control.

7. References

[1] Berners-Lee, T., "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", RFC 1630, June 1994.

[2] Daniel, R., and Mealling, M., "Resolution of Uniform Resource Identifiers using the Domain Name System", RFC 2168, February 1997.

[3] Moats, R., "URN Syntax", RFC 2141, January 1997.

[4] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[5] Sollins, K., "Architectural Principles of Uniform Resource Name Resolution", RFC 2276, January 1998.

[6] Kunze, J., "Functional Recommendations for Internet Resource Locators", RFC 1736, February 1995.

[7] Berners-Lee, T., Masinter, L. and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, December 1994.

8. Authors' Addresses

Michael Mealling
Network Solutions
505 Huntmar Park Drive
Herndon, VA 22070

Phone: (703) 742-0400
Fax: (703) 742-9552
EMail: michaelm@rwhois.net

Ron Daniel
Advanced Computing Lab, MS B287
Los Alamos National Laboratory
Los Alamos, NM, USA, 87545

Phone: (505) 665-0597
Fax: (505) 665-4939
EMail: rdaniel@lanl.gov

9. Full Copyright Statement