Network Working Group
Request for Comments: 2648
Category: Informational

R. Moats
AT&T

August 1999

A URN Namespace for IETF Documents

Status of this Memo

Copyright Notice

Abstract

A system for Uniform Resource Names (URNs) must be capable of supporting new naming systems. As an example of proposing a new namespace, this document proposes the "ietf" namespace. This namespace consists of the RFC family of documents (RFCs, STDs, FYIs, and BCPs) developed by the IETF and published by the RFC Editor, the minutes of working groups (WG) and birds of a feather (BOF) meetings that occur during IETF conferences, and the Internet Drafts published by the Internet Drafts Editor. Both the current URN framework and URN syntax support this namespace.

1. Introduction

This document proposes the "ietf" namespace, which consists of the RFC family of documents (RFCs, STDs, FYIs, and BCPs) developed by the IETF and published by the RFC editor and the minutes of working groups (WG) and birds of a feather (BOF) meetings that occur during IETF conferences.

The namespace specification is for a formal namespace.

2. Specification Template

Namespace ID:

"ietf" requested.

Registration Information:

Registration version number: 1
Registration date: 1999-04-22

Declared registrant of the namespace:

Ryan Moats
jayhawk@att.com
AT&T
15621 Drexel Circle
Omaha, NE 68135-2358

Declaration of structure:

The identifier has the following ABNF [2] specification:

```
NSS = rfc-nss / fyi-nss / std-nss / bcp-nss /
      draft-nss / mtg-nss / other-nss

rfc-nss = "rfc:" 1*DIGIT
fyi-nss = "fyi:" 1*DIGIT
std-nss = "std:" 1*DIGIT
bcp-nss = "bcp:" 1*DIGIT
draft-nss = "id:" string
mtg-nss = "mtg:" string
other-nss = string
  ; beginning with a prefix other than one of those
  ; above for future expansion

string = 1*(DIGIT / ALPHA / "-")
```

If the IESG (or it successor) adds a new document series, this ABNF specification will need to be updated. Further, if a working group or BOF is created that used characters outside the range of this ABNF specification, this specification will need to be updated. Any system intended to resolve names for this namespace should be written with the awareness that this could occur at any time.

Relevant ancillary documentation:

Relevant documentation is in RFC 2648.

Identifier uniqueness considerations:

Because the rfc-editor assigns the RFC number uniquely these URNs are unique. Since the mapping between RFCs and other rfc-editor document series (STDs, FYIs or BCPs) is not necessarily one-to-one, uniqueness of STDs, FYIs and BCPs are defined based on the document mappings maintained by the RFC Editor (the index files "rfc-index.txt", "fyi-index.txt", "bcp-index.txt", "std-index.txt") are defined to be the definitive statement of the assignment of RFC Family URNs in this namespace. The meeting minutes portion of the namespace is guaranteed unique because the URN includes the sequence number of the IETF conference. The document mapping maintained by the Internet Drafts editor ("1id-abstracts.txt") is defined as the definitive statement of the assignment of URNs for the internet draft portion of this namespace.

Identifier persistence considerations:

Persistence of the URNs of this namespace is independent of the mutability of the underlying documents. A URN once assigned will never be reassigned to a different resource; the assignment is persistent and immutable. Immutability of RFCs, STDs, FYIs and BCPs is at the discretion of the RFC Editor. They may be composites of one or more RFCs and the set of RFCs that includes them may change with time. It is important to note that this mutability of some resources is independent of the immutability of URN assignment to a resource.

Process of identifier assignment:

Assignment of URNs from this namespace occurs in three ways. The first is through publication of a new RFC, FYI, STD or BCP is by the RFC Editor. This new document will have a new series number and will therefore define a new URN. The document mappings maintained by the RFC Editor (the index files "rfc-index.txt", "fyi-index.txt", "bcp-index.txt" and "std-index.txt") are defined to be the definitive statement of the assignment of RFC Family URNs in this namespace.

The second way a URN is assigned is through the filing of meeting minutes by a working group or birds of a feather as part of an IETF conference. The list of minutes maintained by the IETF for each working group and conference in the subtree pointed at by the URL ftp://ietf.org/ietf/ is considered the definitive assignment of URNs for working

group or birds of a feather minutes.

The third way a URN is assigned is through the publication of a new internet-draft by the Internet Draft Editor. This draft will have a distinct name (and version number) and therefore defined a new URN. The document mapping maintained by the Internet Drafts editor ("1id-abstracts.txt") is defined as the definitive statement of the assignment of URNs for this portion of the namespace.

Process of identifier resolution:

A mirrored copy of the underlying documentation is required to resolve these URNs. Resolution via HTTP is done by a set of simple Perl cgi-bin scripts presented in Appendix A.

Rules for Lexical Equivalence:

The entire URN is case-insensitive.

Conformance with URN Syntax:

There are no additional characters reserved.

Validation mechanism:

None additional to resolution specified

Scope:

Global.

3. Examples

The following are examples of URNs that a resolver for this namespace can resolve:

```
urn:ietf:rfc:2141
urn:ietf:std:50
urn:ietf:id:ietf-urn-ietf-06
urn:ietf:mtg:41-urn
```

4. Security Considerations

Because this namespace defines no additional reserved characters, it does not add any security considerations beyond those inherent from the existence of the reserved characters from [1]. Further, none of the reserved characters from [1] are used in the definition of the NSS. This means that resolvers for this namespace may be considered "secure" in the sense that any escaping of characters in the NSS MUST result in the resolver indicating that the URN has incorrect syntax.

5. Acknowledgments

Thanks to various members of the URN working group for comments on earlier drafts of this document. The work described in this document is partially supported by the National Science Foundation, Cooperative Agreement NCR-9218179.

6. References

Request For Comments (RFC) and Internet Draft documents are available from numerous mirror sites.

[1] Moats, R., "URN Syntax", RFC 2141, May 1997.

[2] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

7. Author's Address

Ryan Moats
AT&T
15621 Drexel Circle
Omaha, NE 68135-2358
USA

EMail: jayhawk@att.com

Appendix A. Example Resolution Scripts

The following scripts are examples that can be used for resolving URNs in this namespace.

A.1 I2C

```
#!/usr/local/bin/perl

use strict;

#
# this is a URN 2 URC resolver for the ietf namespace
#

my(%cite) = (
  bcp => "/ftp/rfc/bcp-index.txt",
  fyi => "/ftp/fyi/fyi-index.txt",
  id => "/ftp/internet-drafts/1id-abstracts.txt",
  rfc => "/ftp/rfc/rfc-index.txt",
  std => "/ftp/std/std-index.txt"
);
my(%number2date) = (
  44 => "99mar",
  43 => "98dec", 42 => "98aug", 41 => "98apr",
  40 => "97dec", 39 => "97aug", 38 => "97apr",
  37 => "96dec", 36 => "96jun", 35 => "96mar",
  34 => "95dec", 33 => "95jul", 32 => "95apr",
  31 => "94dec", 30 => "94jul", 29 => "94mar",
  28 => "93nov", 27 => "93jul", 26 => "93mar",
  25 => "92nov", 24 => "92jul", 23 => "92mar",
  22 => "91nov", 21 => "91jul", 20 => "91mar",
  19 => "90dec" );

my($wgpath) = "/ftp/ietf";
my($urn) = $ENV{'QUERY_STRING'};
my($host) = $ENV{'SERVER_NAME'}; #get my host name for ftp: URLs
my($accept) = $ENV{'HTTP_ACCEPT'}; #this is the "Accept:" HTTP header

(&resolveid($1), exit) if ($urn =~ /urn:ietf:id:(\S+)/i);
(&resolverfc($1, $2), exit) if ($urn =~ /urn:ietf:(\w*):(\d*)/i);
(&resolvemtg($1, $2), exit) if ($urn =~ /urn:ietf:mtg:(\d*)-(\w*)/i);
&urn_error("400 Bad Request\n");

sub resolvemtg {
  my($ietfnum, $sesnam) = @_;
  &urn_error("404 Not Found\n") if (!defined $number2date{$ietfnum});
  my($date)=$number2date{$ietfnum};
```

```
  my($link)="$wgpath/$sesnam/$sesnam-minutes-$date.txt";
  if (-f $link) {
    print "Status:  200 OK\r\n";
    print "Content-type: text/html\r\n\r\n";
    print "<HTML>\n<TITLE>Citation for $urn</TITLE>\n";
    print "<BODY>\n";
    print "<H1><A HREF=\"$link\">$urn</A>:</H1>\n";
    print "Minutes of the $sesnam working group from the "
          . &end($ietfnum) . " IETF";
    print "</BODY>\n</HTML>\n";
    return;
  }
  my($link)="$wgpath/$date/$sesnam-minutes-$date.txt";
  if (-f $link) {
    print "Status:  200 OK\r\n";
    print "Content-type: text/html\r\n\r\n";
    print "<HTML>\n<TITLE>Citation for $urn</TITLE>\n";
    print "<BODY>\n";
    print "<H1><A HREF=\"$link\">$urn</A>:</H1>\n";
    print "Minutes of the $sesnam working group from the "
           . &end($ietfnum) . " IETF";
    print "</BODY>\n</HTML>\n";
    return;
  }
  &urn_error("404 Not Found\n");
}

sub end {
  my($inarg)=@_;
  return $inarg . "st" if ($inarg =~ /1$/);
  return $inarg . "nd" if ($inarg =~ /2$/);
  return $inarg . "rd" if ($inarg =~ /3$/);
  return $inarg . "th";
}

sub resolverfc {
  my($flag,@bib,$i,$k,$j,$done,@ref);
  my($l,$link);
  my($scheme, $value) = @_;
  $scheme =~ tr/A-Z/a-z/;
  if (!defined $cite{$scheme}) {
    &urn_error("404 Not Found\n");
  }

  $flag = 0;
  open(INPUT, "$cite{$scheme}");
  while (<INPUT>) {
    $flag = 1 if (/^0*$value /);
```

```
    if ($flag == 1) {
      last if (/^$/);
      chop;
      push @bib,$_;
    }
  }

  if ($scheme ne "rfc") {
    print "Status:  200 OK\r\n";
    print "Content-type: text/html\r\n\r\n";
    $bib[0] =~ s/^[0-9]*\s*/<B>/;
    for ($i=0; $i<=$#bib; $i+=1) {
      last if ($bib[$i] =~ s/\./.<\/B>/);
    }
    for ($i=0;$i<=$#bib;$i+=1) {
      $k=$bib[$i];
      while ($k =~ /(fyi|std|rfc|bcp)([0-9]+)(.*)/i) {
        push @ref,"$1$2";
        $k=$3;
      }
      $done="";
      foreach $j (@ref) {
        next if ($done =~ $j);
        $done .= "$j ";
        $l = $j;
        $l =~ tr/A-Z/a-z/;
        $link=&make_link("$l");
        $bib[$i] =~ s/$j/<A HREF="$link">$j<\/A>/g;
      }
    }
    print "<HTML>\n<TITLE>Citation for $urn</TITLE>\n";
    print "<BODY>\n";
    $link=&make_link("$scheme$value");
    print "<H1><A HREF=\"$link\">$scheme$value</A>:</H1>\n";
    foreach $i (@bib) {
      print "$i\n";
    }
    print "</BODY>\n</HTML>\n";
  } else {
    print "Status:  200 OK\r\n";
    print "Content-type: text/html\r\n\r\n";
    $bib[0] =~ s/^[0-9]*\s*//;
    $j=0;
    for ($i=0; $i<=$#bib; $i+=1) {
      $j += ($bib[$i] =~ s/, "/, <B>"/);
      $j += ($bib[$i] =~ s/",/"<\/B>,/);
    }
    for ($i=0;$i<=$#bib;$i+=1) {
```

```
      $k=$bib[$i];
      while ($k =~ /(fyi\s|std\s|rfc|bcp)([0-9]+)(.*)/i) {
        push @ref,"$1$2";
        $k=$3;
      }
      $done="";
      foreach $j (@ref) {
        next if ($done =~ $j);
        $done .= "$j ";
        $l = $j;
        $l =~ s/\s//g;
        $l =~ tr/A-Z/a-z/;
        $link=&make_link("$l");
        $bib[$i] =~ s/$j/<A HREF="$link">$j<\/A>/g;
      }
    }
    print "<HTML>\n<TITLE>Citation for $urn</TITLE>\n";
    print "<BODY>\n";
    $link=&make_link("$scheme$value");
    print "<H1><A HREF=\"$link\">$scheme$value</A>:</H1>\n";
    foreach $i (@bib) {
      print "$i\n";
    }
    print "</BODY>\n</HTML>\n";
  }
}

sub make_link {
  my($sc);
  my($inarg)=@_;
  ($sc=$1) if ($inarg =~ /([a-z]*)/);
  return "/$sc/$inarg.ps" if (-e "/ftp/$sc/$inarg.ps");
  return "/$sc/$inarg.html" if (-e "/ftp/$sc/$inarg.html");
  return "/$sc/$inarg.txt";
}

sub urn_error {
  my($code) = @_; #store failure code here...

  print "Status:  $code";
  print "Content-type: text/html\n\n<HTML>\n";
  print "<head><title>URN Resolution: I2C $code</title></head>\n";
  print "<BODY>\n";
  print "<h1>URN to URC resolution failed for the URN:</h1>\n";
  print "<hr><h3>$urn</h3>\n";
  print "</body>\n";
  print "</html>\n";
  exit;
```

```
};

sub resolveid {
  my($flag,@bib,$i,$k,$j,$count,@ref);
  my($l,$link, $hdr, $done);
  my($value) = @_;
  my($scheme) = "id";

  open(INPUT, "$cite{$scheme}");
  while (<INPUT>) {
#
# capture record
#
    if ($flag == 1 || /^\s+\"/) {
      push @bib,$_;
      ($hdr = -1, $count = 0, $flag = 1) if (/^\s+\"/);
      $count++ if (/^\s+$/);
    }
    if ($count == 1) {
      $hdr = $#bib if ($hdr == -1);
    }
    if ($count == 2) {
      for ($i=0; $i<=$hdr; $i+=1) {
            if ($bib[$i] =~ /<(.*)>/) {
              $l = $1;
              if ($l eq "draft-$value.txt" || $l eq "draft-$value.ps") {
                print "Status:  200 OK\r\n";
                print "Content-type: text/html\r\n\r\n";
                print "<HTML>\n<TITLE>Citation for $urn</TITLE>\n";
                print "<BODY>\n";
                print "<a
href=\"http://blackhole.vip.att.net/internet-drafts/$l\">$l</a>:\n";
                print "<pre>\n";
                foreach $i (@bib) {
                  print "$i";
                }
                print "</pre>\n";
                print "</BODY>\n</HTML>\n";
                exit;
              }
            }
      }
      $flag = 0;
      @bib = ();
    }
  }
  &urn_error("404 Not Found\n");
}
```

A.2 I2L

```
#!/usr/local/bin/perl

use strict;

#
# this is a URN 2 URL resolver for the ietf namespace
#

my(%pathbase) = (
  rfc => "rfc/rfc",
  fyi => "fyi/fyi",
  std => "std/std",
  bcp => "bcp/bcp",
  id => "internet-drafts/draft-"
);

my(%number2date) = (
  44 => "99mar",
  43 => "98dec", 42 => "98aug", 41 => "98apr",
  40 => "97dec", 39 => "97aug", 38 => "97apr",
  37 => "96dec", 36 => "96jun", 35 => "96mar",
  34 => "95dec", 33 => "95jul", 32 => "95apr",
  31 => "94dec", 30 => "94jul", 29 => "94mar",
  28 => "93nov", 27 => "93jul", 26 => "93mar",
  25 => "92nov", 24 => "92jul", 23 => "92mar",
  22 => "91nov", 21 => "91jul", 20 => "91mar",
  19 => "90dec" );

my($wgpath) = "/ftp/ietf";
my($urn) = $ENV{'QUERY_STRING'};
my($host) = $ENV{'SERVER_NAME'}; #get my host name for ftp: URLs
my($accept) = $ENV{'HTTP_ACCEPT'}; #this is the "Accept:" HTTP header

(&resolveid($1), exit) if ($urn =~ /urn:ietf:id:(\S+)/i);
(&resolverfc($1, $2), exit) if ($urn =~ /urn:ietf:(\w*):(\d*)/i);
(&resolvemtg($1, $2), exit) if ($urn =~ /urn:ietf:mtg:(\d*)-(\w*)/i);
&urn_error("400 Bad Request\n");

sub resolvemtg {
  my($ietfnum, $sesnam) = @_;
  &urn_error("404 Not Found\n") if (!defined $number2date{$ietfnum});
  my($date)=$number2date{$ietfnum};
  my($link)="$wgpath/$sesnam/$sesnam-minutes-$date.txt";
  if (-f $link) {
    print "Status:  302 Moved temporarily\n";
    print "Location: $link\n";
```

```
    return;
  }
  my($link)="$wgpath/$date/$sesnam-minutes-$date.txt";
  if (-f $link) {
    print "Status:  302 Moved temporarily\n";
    print "Location: $link\n";
    return;
  }
  &urn_error("404 Not Found\n");
}

sub end {
  my($inarg)=@_;
  return $inarg . "st" if ($inarg =~ /1$/);
  return $inarg . "nd" if ($inarg =~ /2$/);
  return $inarg . "rd" if ($inarg =~ /3$/);
  return $inarg . "th";
}

sub resolverfc {
  my($flag,@bib,$i,$k,$j,$done,@ref);
  my($l,$link);
  my($scheme, $value) = @_;
  $scheme =~ tr/A-Z/a-z/;
  &urn_error("404 Not Found\n")if (!defined $pathbase{$scheme});
  my($txttry)="/ftp/$pathbase{$scheme}$value.txt";
  my($pstry)="/ftp/$pathbase{$scheme}$value.ps";
  my($htmltry)="/ftp/$pathbase{$scheme}$value.html";
MIME_SWITCH: {
    if ($accept =~ /application\/postscript/ && -f $pstry) {
      print "Status:  302 Moved temporarily\n";
      print "Location: http://$host/$pathbase{$scheme}$value.ps\n\n";
      last MIME_SWITCH;
    }
    if ($accept =~ /text\/html/ && -f $htmltry) {
      print "Status:  302 Moved temporarily0;
      print "Location: http://$host/$pathbase{$scheme}$value.html\n\n";
      last MIME_SWITCH;
    }
    if ($accept =~ /\*\/\*|text\/plain/ && -f $txttry) {
      print "Status:  302 Moved temporarily\n";
      print "Location: http://$host/$pathbase{$scheme}$value.txt\n\n";
      last MIME_SWITCH;
    }
    &urn_error("404 Not Found\n");
  }
}
```

```
sub urn_error {
  my($code) = @_; #store failure code here...

  print "Status:  $code";
  print "Content-type: text/html\n\n<HTML>\n";
  print "<head><title>URN Resolution: I2L $code</title></head>\n";
  print "<BODY>\n";
  print "<h1>URN to URL resolution failed for the URN:</h1>\n";
  print "<hr><h3>$urn</h3>\n";
  print "</body>\n";
  print "</html>\n";
  exit;
}

sub resolveid {
  my($flag,@bib,$i,$k,$j,$done,@ref);
  my($l,$link);
  my($scheme) = "id";
  my($value) = @_;
  $scheme =~ tr/A-Z/a-z/;
  &urn_error("404 Not Found\n")if (!defined $pathbase{$scheme});
  my($txttry)="/ftp/$pathbase{$scheme}$value.txt";
  my($pstry)="/ftp/$pathbase{$scheme}$value.ps";
  my($htmltry)="/ftp/$pathbase{$scheme}$value.html";
MIME_SWITCH: {
    if ($accept =~ /application\/postscript/ && -f $pstry) {
      print "Status:  302 Moved temporarily\n";
      print "Location: http://$host/$pathbase{$scheme}$value.ps\n\n";
      last MIME_SWITCH;
    }
    if ($accept =~ /text\/html/ && -f $htmltry) {
      print "Status:  302 Moved temporarily\n";
      print "Location: http://$host/$pathbase{$scheme}$value.html\n\n";
      last MIME_SWITCH;
    }
    if ($accept =~ /\*\/\*|text\/plain/ && -f $txttry) {
      print "Status:  302 Moved temporarily\n";
      print "Location: http://$host/$pathbase{$scheme}$value.txt\n\n";
      last MIME_SWITCH;
    }
    &urn_error("404 Not Found\n");
  }
}
```

A.3 I2Ls

```
#!/usr/local/bin/perl

use strict;

#
# this is a URN 2 URLs resolver for the ietf namespace
#

my(@urls);

my(%pathbase) = (
  rfc => "rfc/rfc",
  fyi => "fyi/fyi",
  std => "std/std",
  bcp => "bcp/bcp",
  id => "internet-drafts/draft-"
);

my(%number2date) = (
  44 => "99mar",
  43 => "98dec", 42 => "98aug", 41 => "98apr",
  40 => "97dec", 39 => "97aug", 38 => "97apr",
  37 => "96dec", 36 => "96jun", 35 => "96mar",
  34 => "95dec", 33 => "95jul", 32 => "95apr",
  31 => "94dec", 30 => "94jul", 29 => "94mar",
  28 => "93nov", 27 => "93jul", 26 => "93mar",
  25 => "92nov", 24 => "92jul", 23 => "92mar",
  22 => "91nov", 21 => "91jul", 20 => "91mar",
  19 => "90dec" );

my($wgpath) = "/ftp/ietf";
my($urn) = $ENV{'QUERY_STRING'};
my($host) = $ENV{'SERVER_NAME'}; #get my host name for ftp: URLs
my($accept) = $ENV{'HTTP_ACCEPT'}; #this is the "Accept:" HTTP header

(&resolveid($1), exit) if ($urn =~ /urn:ietf:id:(\S+)/i);
(&resolve1($1, $2), exit) if ($urn =~ /urn:ietf:(\w*):(\d*)/i);
(&resolve2($1, $2), exit) if ($urn =~ /urn:ietf:mtg:(\d*)-(\w*)/i);
&urn_error("400 Bad Request\n");

sub resolve2 {
  my($ietfnum, $sesnam) = @_;
  &urn_error("404 Not Found\n") if (!defined $number2date{$ietfnum});
  my($date)=$number2date{$ietfnum};
  my($link)="$wgpath/$sesnam/$sesnam-minutes-$date.txt";
  if (-f $link) {
```

```
    $link=~s/^\/ftp\///;
    my($ftplink)="ftp://$host/$link";
    my($httplink)="http://$host/$link";
    my($glink)="gopher://$host:70/0/$link";

    if ($accept =~ /text\/uri-list/) { #look for text/uri-list,
        otherwise text/html
      print "Status: 200 OK\n";
      print "Content-type: text/uri-list\n\n\n";
      print "#$urn\n";
      print "$ftplink\n";
      print "$httplink\n";
      print "$glink\n";
    }
    if ($accept =~ /\*\/\*|text\/html/) {
      print "Status: 200 OK\n";
      print "Content-type: text/html\n\n<HTML>\n";
      print "<head><title>URN Resolution: I2Ls</title></head>\n";
      print "<BODY>\n";
      print "<h1>URN $urn resolves to the following URLs:</h1>\n";
      print "<hr><ul>\n";
      print "<a href=\"$ftplink\">$ftplink</a>\n";
      print "<a href=\"$httplink\">$httplink</a>\n";
      print "<a href=\"$glink\">$glink</a>\n";
      print "</UL>\n</body>\n</HTML>\n";
    }
    return;
  }
  my($link)="$wgpath/$date/$sesnam-minutes-$date.txt";
  if (-f $link) {
    $link=~s/^\/ftp\///;
    my($ftplink)="ftp://$host/$link";
    my($httplink)="http://$host/$link";
    my($glink)="gopher://$host:70/0/$link";
    if ($accept =~ /text\/uri-list/) { #look for text/uri-list,
        otherwise text/html
      print "Status: 200 OK\n";
      print "Content-type: text/uri-list\n\n\n";
      print "#$urn\n";
      print "$ftplink\n";
      print "$httplink\n";
      print "$glink\n";
    }
    if ($accept =~ /\*\/\*|text\/html/) {
      print "Status: 200 OK\n";
      print "Content-type: text/html\n\n<HTML>\n";
      print "<head><title>URN Resolution: I2Ls</title></head>\n";
      print "<BODY>\n";
```

```
      print "<h1>URN $urn resolves to the following URLs:</h1>\n";
      print "<hr><ul>\n";
      print "<a href=\"$ftplink\">$ftplink</a>\n";
      print "<a href=\"$httplink\">$httplink</a>\n";
      print "<a href=\"$glink\">$glink</a>\n";
      print "</UL>\n</body>\n</HTML>\n";
    }

    return;
  }
  &urn_error("404 Not Found\n");
}

sub resolve1 {
  my($flag,@bib,$i,$k,$j,$done,@ref);
  my($l,$link);
  my($scheme, $value) = @_;
  $scheme =~ tr/A-Z/a-z/;
  &urn_error("404 Not Found\n")if (!defined $pathbase{$scheme});
  my($try)="/ftp/$pathbase{$scheme}$value.txt";
  if (-f $try) {
    push(@urls, "http://$host/$pathbase{$scheme}$value.txt");
    push(@urls, "ftp://$host/$pathbase{$scheme}$value.txt");
    push(@urls, "gopher://$host:70/0/$pathbase{$scheme}$value.txt");
  }
  $try="/ftp/$pathbase{$scheme}$value.ps";
  if (-f $try) {
    push(@urls, "http://$host/$pathbase{$scheme}$value.ps");
    push(@urls, "ftp://$host/$pathbase{$scheme}$value.ps");
    push(@urls, "gopher://$host:70/0/$pathbase{$scheme}$value.ps");
  }
  $try="/ftp/$pathbase{$scheme}$value.html";
  if (-f $try) {
    push(@urls, "http://$host/$pathbase{$scheme}$value.html");
    push(@urls, "ftp://$host/$pathbase{$scheme}$value.html");
  }

  &urn_error("404 Not Found\n") if ($#urls == -1);

MIME_SWITCH: {
    if ($accept =~ /text\/uri-list/) { #look for text/uri-list,
        otherwise text/html
      print "Status: 200 OK\n";
      print "Content-type: text/uri-list\n\n\n";
      print "#$urn\n";
      foreach $i (@urls) {
        print "$i\n";
      }
```

```
      last MIME_SWITCH;
    }
    if ($accept =~ /\*\/\*|text\/html/) {
      print "Status: 200 OK\n";
      print "Content-type: text/html\n\n<HTML>\n";
      print "<head><title>URN Resolution: I2Ls</title></head>\n";
      print "<BODY>\n";
      print "<h1>URN $urn resolves to the following URLs:</h1>\n";
      print "<hr><ul>\n";
      foreach $i (@urls) {
        print "<LI><A HREF=\"$i\">$i</A>\n";
      }
      print "</UL>\n</body>\n</HTML>\n";
      last MIME_SWITCH;
    }
  }
}

sub urn_error {
  my($code) = @_; #store failure code here...

  print "Status: $code";
  print "Content-type: text/html\n\n<HTML>\n";
  print "<head><title>URN Resolution: I2L $code</title></head>\n";
  print "<BODY>\n";
  print "<h1>URN to URL resolution failed for the URN:</h1>\n";
  print "<hr><h3>$urn</h3>\n";
  print "</body>\n";
  print "</html>\n";
  exit;
}

sub resolveid {
  my($flag,@bib,$i,$k,$j,$done,@ref);
  my($l,$link);
  my($value) = @_;
  my($scheme) = "id";
  $scheme =~ tr/A-Z/a-z/;
  &urn_error("404 Not Found\n")if (!defined $pathbase{$scheme});
  my($try)="/ftp/$pathbase{$scheme}$value.txt";
  if (-f $try) {
    push(@urls, "http://$host/$pathbase{$scheme}$value.txt");
    push(@urls, "ftp://$host/$pathbase{$scheme}$value.txt");
    push(@urls, "gopher://$host:70/0/$pathbase{$scheme}$value.txt");
  }
  $try="/ftp/$pathbase{$scheme}$value.ps";
  if (-f $try) {
    push(@urls, "http://$host/$pathbase{$scheme}$value.ps");
```

```
    push(@urls, "ftp://$host/$pathbase{$scheme}$value.ps");
    push(@urls, "gopher://$host:70/0/$pathbase{$scheme}$value.ps");
  }
  $try="/ftp/$pathbase{$scheme}$value.html";
  if (-f $try) {
    push(@urls, "http://$host/$pathbase{$scheme}$value.html");
    push(@urls, "ftp://$host/$pathbase{$scheme}$value.html");
  }

  &urn_error("404 Not Found\n") if ($#urls == -1);

MIME_SWITCH: {
    if ($accept =~ /text\/uri-list/) { #look for text/uri-list,
        otherwise text/html
      print "Status: 200 OK\n";
      print "Content-type: text/uri-list\n\n\n";
      print "#$urn\n";
      foreach $i (@urls) {
        print "$i\n";
      }
      last MIME_SWITCH;
    }
    if ($accept =~ /\*\/\*|text\/html/) {
      print "Status: 200 OK\n";
      print "Content-type: text/html\n\n<HTML>\n";
      print "<head><title>URN Resolution: I2Ls</title></head>\n";
      print "<BODY>\n";
      print "<h1>URN $urn resolves to the following URLs:</h1>\n";
      print "<hr><ul>\n";
      foreach $i (@urls) {
        print "<LI><A HREF=\"$i\">$i</A>\n";
      }
      print "</UL>\n</body>\n</HTML>\n";
      last MIME_SWITCH;
    }
  }
}
```

A.4 I2Ns

```
#!/usr/local/bin/perl

use strict;

#
# this is a URN 2 URNs resolver for the ietf namespace
#
```

```
my(%cite) = (
  rfc => "/ftp/rfc/rfc-index.txt",
  fyi => "/ftp/fyi/fyi-index.txt",
  std => "/ftp/std/std-index.txt",
  bcp => "/ftp/rfc/bcp-index.txt"
);

my(%number2date) = (
  44 => "99mar",
  43 => "98dec", 42 => "98aug", 41 => "98apr",
  40 => "97dec", 39 => "97aug", 38 => "97apr",
  37 => "96dec", 36 => "96jun", 35 => "96mar",
  34 => "95dec", 33 => "95jul", 32 => "95apr",
  31 => "94dec", 30 => "94jul", 29 => "94mar",
  28 => "93nov", 27 => "93jul", 26 => "93mar",
  25 => "92nov", 24 => "92jul", 23 => "92mar",
  22 => "91nov", 21 => "91jul", 20 => "91mar",
  19 => "90dec" );

my($wgpath) = "/ftp/ietf";
my($urn) = $ENV{'QUERY_STRING'};
my($host) = $ENV{'SERVER_NAME'}; #get my host name for ftp: URLs
my($port) = $ENV={'SERVER_PORT'};
my($accept) = $ENV{'HTTP_ACCEPT'}; #this is the "Accept:" HTTP header

(&resolve1($1, $2), exit) if ($urn =~ /urn:ietf:(\w*):(\d*)/i);
(&resolve2($1, $2), exit) if ($urn =~ /urn:ietf:mtg:(\d*)-(\w*)/i);
&urn_error("400 Bad Request\n");

sub resolve2 {
  my($ietfnum, $sesnam) = @_;
  &urn_error("404 Not Found\n") if (!defined $number2date{$ietfnum});
  my($date)=$number2date{$ietfnum};
  my($link)="$wgpath/$sesnam/$sesnam-minutes-$date.txt";
  if (-f $link) {
    if ($accept =~ /text\/uri-list/) {
        print "Status: 200 OK\n";
        print "Content-type: text/uri-list\n\n\n";
        print "#$urn\n";
        return;
    }
    if ($accept =~ /\*\/\*|text]\/html/) {
      print "Status: 200 OK\n";
      print "Content-type: text/html\n\n<HTML>\n";
      print "<head><title>URN Resolution: I2Ns</title></head>\n";
      print "<BODY>\n";
      print "<h1>URN $urn resolves to the following URNs:</h1>\n";
      print "<hr><ul>\n";
```

```
      print "</UL>\n</body>\n</HTML>\n";
      return;
    }
  }
  my($link)="$wgpath/$date/$sesnam-minutes-$date.txt";
  if (-f $link) {
    if ($accept =~ /text\/uri-list/) {
        print "Status: 200 OK\n";
        print "Content-type: text/uri-list\n\n\n";
        print "#$urn\n";
        return;
    }
    if ($accept =~ /\*\/\*|text\/html/) {
        print "Status: 200 OK\n";
        print "Content-type: text/html\n\n<HTML>\n";
        print "<head><title>URN Resolution: I2Ns</title></head>\n";
        print "<BODY>\n";
        print "<h1>URN $urn resolves to the following URNs:</h1>\n";
        print "<hr><ul>\n";
        print "</UL>\n</body>\n</HTML>\n";
        return;
    }
  }
  &urn_error("404 Not Found\n");
}

sub end {
  my($inarg)=@_;
  return $inarg . "st" if ($inarg =~ /1$/);
  return $inarg . "nd" if ($inarg =~ /2$/);
  return $inarg . "rd" if ($inarg =~ /3$/);
  return $inarg . "th";
}

sub resolve1 {
  my($flag,@bib,$i,$k,$j,$done,@ref);
  my($l,$link);
  my($scheme, $value) = @_;
  $scheme =~ tr/A-Z/a-z/;
  if (!defined $cite{$scheme}) {
    &urn_error("404 Not Found\n");
  }

  $flag = 0;
  open(INPUT, "$cite{$scheme}");
  while (<INPUT>) {
    $flag = 1 if (/^0*$value /);
    if ($flag == 1) {
```

```
      last if (/^$/);
      chop;
      push @bib,$_;
    }
  }

  $k=join " ",@bib;
  while ($k =~ /(\S*)\s*(fyi|std|rfc|bcp)\s*([0-9]+)(.*)/i) {
    $k=$4;
    $a=$2; $b=$3;
    if (($a ne $scheme || $b ne $value) && ($1 !~ /obso/i)){
      $a =~ tr/A-Z/a-z/;
      $b =~ s/^0*//;
      push @ref,"urn:ietf:$a:$b";
    }
  }

MIME_SWITCH: {
    if ($accept =~ /text\/uri-list/) {
        print "Status: 200 OK\n";
        print "Content-type: text/uri-list\n\n\n";
        print "#$urn\n";
        foreach $i (@ref) {
            print "$i\n";
        }
        last MIME_SWITCH;
    }
  if ($accept =~ /\*\/\*|text\/html/) {
    print "Status: 200 OK\n";
    print "Content-type: text/html\n\n<HTML>\n";
    print "<head><title>URN Resolution: I2Ns</title></head>\n";
    print "<BODY>\n";
    print "<h1>URN $urn resolves to the following URNs:</h1>\n";
    print "<hr><ul>\n";
        foreach $i (@ref) {
            print "<li>$i: Click to resolve using\n";
            print "<a
href=\"http://$host:$port/uri-res/I2C?$i\">I2C</a>,\n";
            print "<a
href=\"http://$host:$port/uri-res/I2L?$i\">I2L</a>,\n";
            print "<a
href=\"http://$host:$port/uri-res/I2Ls?$i\">I2Ls</a>,\n";
            print "<a
href=\"http://$host:$port/uri-res/I2R?$i\">I2R</a>,\n";
            print "<a
href=\"http://$host:$port/uri-res/I2Rs?$i\">I2Rs</a>\n";
        }
    print "</UL>\n</body>\n</HTML>\n";
```

```
  }
}
}

sub make_link {
  my($sc);
  my($inarg)=@_;
  ($sc=$1) if ($inarg =~ /([a-z]*)/);
  return "/$sc/$inarg.ps" if (-e "/ftp/$sc/$inarg.ps");
  return "/$sc/$inarg.html" if (-e "/ftp/$sc/$inarg.html");
  return "/$sc/$inarg.txt";
}

sub urn_error {
  my($code) = @_; #store failure code here...

  print "Status: $code";
  print "Content-type: text/html\n\n<HTML>\n";
  print "<head><title>URN Resolution: I2Ns $code</title></head>\n";
  print "<BODY>\n";
  print "<h1>URN to URN resolution failed for the URN:</h1>\n";
  print "<hr><h3>$urn</h3>\n";
  print "</body>\n";
  print "</html>\n";
  exit;
};
```

A.5 I2R

```
#!/usr/local/bin/perl

use strict;

#
# this is a URN 2 resource resolver for the ietf namespace
#

my(%pathbase) = (
  rfc => "rfc/rfc",
  fyi => "fyi/fyi",
  std => "std/std",
  bcp => "bcp/bcp",
  id => "internet-drafts/draft-"
);

my(%number2date) = (
  44 => "99mar",
  43 => "98dec", 42 => "98aug", 41 => "98apr",
```

```
  40 => "97dec", 39 => "97aug", 38 => "97apr",
  37 => "96dec", 36 => "96jun", 35 => "96mar",
  34 => "95dec", 33 => "95jul", 32 => "95apr",
  31 => "94dec", 30 => "94jul", 29 => "94mar",
  28 => "93nov", 27 => "93jul", 26 => "93mar",
  25 => "92nov", 24 => "92jul", 23 => "92mar",
  22 => "91nov", 21 => "91jul", 20 => "91mar",
  19 => "90dec" );

my($wgpath) = "/ftp/ietf";
my($urn) = $ENV{'QUERY_STRING'};
my($host) = $ENV{'SERVER_NAME'}; #get my host name for ftp: URLs
my($accept) = $ENV{'HTTP_ACCEPT'}; #this is the "Accept:" HTTP header

print "$urn\n";
(&resolveid($1), exit) if ($urn =~ /urn:ietf:id:(\S+)/i);
(&resolve1($1, $2), exit) if ($urn =~ /urn:ietf:(\w*):(\d*)/i);
(&resolve2($1, $2), exit) if ($urn =~ /urn:ietf:mtg:(\d*)-(\w*)/i);
&urn_error("400 Bad Request\n");

sub resolve2 {
  my($ietfnum, $sesnam) = @_;
  &urn_error("404 Not Found\n") if (!defined $number2date{$ietfnum});
  my($date)=$number2date{$ietfnum};
  my($link)="$wgpath/$sesnam/$sesnam-minutes-$date.txt";
  if (-f $link) {
      print "Status:  200 OK\n";
      print "Content-type: text/plain\n\n";
      open(FILE, "$link");
      while (<FILE>) {
          print $_;
      }
      close FILE;
    return;
  }
  my($link)="$wgpath/$date/$sesnam-minutes-$date.txt";
  if (-f $link) {
      print "Status:  200 OK\n";
      print "Content-type: text/plain\n\n";
      open(FILE, "$link");
      while (<FILE>) {
          print $_;
      }
      close FILE;
    return;
  }
  &urn_error("404 Not Found\n");
}
```

```
sub end {
  my($inarg)=@_;
  return $inarg . "st" if ($inarg =~ /1$/);
  return $inarg . "nd" if ($inarg =~ /2$/);
  return $inarg . "rd" if ($inarg =~ /3$/);
  return $inarg . "th";
}

sub resolve1 {
  my($flag,@bib,$i,$k,$j,$done,@ref);
  my($l,$link);
  my($scheme, $value) = @_;
  $scheme =~ tr/A-Z/a-z/;
  &urn_error("404 Not Found\n")if (!defined $pathbase{$scheme});
  my($txttry)="/ftp/$pathbase{$scheme}$value.txt";
  my($pstry)="/ftp/$pathbase{$scheme}$value.ps";
  my($htmltry)="/ftp/$pathbase{$scheme}$value.html";
MIME_SWITCH: {
    if ($accept =~ /application\/postscript/ && -f $pstry) {
      print "Status:  200 OK\n";
      print "Content-type: application/postscript\n\n";
      open(FILE, "$pstry");
      while (<FILE>) {
          print $_;
      }
      close FILE;
      last MIME_SWITCH;
    }
    if ($accept =~ /text\/html/ && -f $htmltry) {
        print "Status:  200 OK\n";
        print "Content-type: text/html\n\n";
        open(FILE, "$htmltry");
        while (<FILE>) {
            print $_;
        }
        close FILE;
        last MIME_SWITCH;
    }
    if ($accept =~ /\*\/\*|text\/plain/ && -f $txttry) {
        print "Status:  200 OK\n";
        print "Content-type: text/plain\n\n";
        open(FILE, "$txttry");
        while (<FILE>) {
            print $_;
        }
        close FILE;
        last MIME_SWITCH;
    }
```

```
    &urn_error("404 Not Found\n");
  }
}

sub resolveid {
  my($flag,@bib,$i,$k,$j,$done,@ref);
  my($l,$link);
  my($scheme) = "id";
  my($value) = @_;
  $scheme =~ tr/A-Z/a-z/;
  &urn_error("404 Not Found\n")if (!defined $pathbase{$scheme});
  my($txttry)="/ftp/$pathbase{$scheme}$value.txt";
  my($pstry)="/ftp/$pathbase{$scheme}$value.ps";
  my($htmltry)="/ftp/$pathbase{$scheme}$value.html";
MIME_SWITCH: {
    if ($accept =~ /application\/postscript/ && -f $pstry) {
      print "Status:  200 OK\n";
      print "Content-type: application/postscript\n\n";
      open(FILE, "$pstry");
      while (<FILE>) {
          print $_;
      }
      close FILE;
      last MIME_SWITCH;
    }
    if ($accept =~ /text\/html/ && -f $htmltry) {
        print "Status:  200 OK\n";
        print "Content-type: text/html\n\n";
        open(FILE, "$htmltry");
        while (<FILE>) {
            print $_;
        }
        close FILE;
        last MIME_SWITCH;
    }
    if ($accept =~ /\*\/\*|text\/plain/ && -f $txttry) {
        print "Status:  200 OK\n";
        print "Content-type: text/plain\n\n";
        open(FILE, "$txttry");
        while (<FILE>) {
            print $_;
        }
        close FILE;
        last MIME_SWITCH;
    }
    &urn_error("404 Not Found\n");
  }
}
```

```
sub urn_error {
  my($code) = @_; #store failure code here...

  print "Status:  $code";
  print "Content-type: text/html\n\n<HTML>\n";
  print "<head><title>URN Resolution: I2R $code</title></head>\n";
  print "<BODY>\n";
  print "<h1>URN to URL resolution failed for the URN:</h1>\n";
  print "<hr><h3>$urn</h3>\n";
  print "</body>\n";
  print "</html>\n";
  exit;
}
```

A.6 I2Rs

```
#!/usr/local/bin/perl

use strict;

#
# this is a URN 2 resources resolver for the ietf namespace
#

my(@urls);

my(%pathbase) = (
  rfc => "rfc/rfc",
  fyi => "fyi/fyi",
  std => "std/std",
  bcp => "bcp/bcp",
  id => "internet-drafts/draft-"
);

my(%number2date) = (
  44 => "99mar",
  43 => "98dec", 42 => "98aug", 41 => "98apr",
  40 => "97dec", 39 => "97aug", 38 => "97apr",
  37 => "96dec", 36 => "96jun", 35 => "96mar",
  34 => "95dec", 33 => "95jul", 32 => "95apr",
  31 => "94dec", 30 => "94jul", 29 => "94mar",
  28 => "93nov", 27 => "93jul", 26 => "93mar",
  25 => "92nov", 24 => "92jul", 23 => "92mar",
  22 => "91nov", 21 => "91jul", 20 => "91mar",
  19 => "90dec" );

my($wgpath) = "/ftp/ietf";
my($urn) = $ENV{'QUERY_STRING'};
```

```
my($host) = $ENV{'SERVER_NAME'}; #get my host name for ftp: URLs
my($accept) = $ENV{'HTTP_ACCEPT'}; #this is the "Accept:" HTTP header

(&resolveid($1), exit) if ($urn =~ /urn:ietf:id:(\s*)/i);
(&resolve1($1, $2), exit) if ($urn =~ /urn:ietf:(\w*):(\d*)/i);
(&resolve2($1, $2), exit) if ($urn =~ /urn:ietf:mtg:(\d*)-(\w*)/i);
&urn_error("400 Bad Request\n");

sub resolve2 {
  my($ietfnum, $sesnam) = @_;
  my(@vers,$i);
  &urn_error("404 Not Found\n") if (!defined $number2date{$ietfnum});
  my($date)=$number2date{$ietfnum};
  my($link)="$wgpath/$sesnam/$sesnam-minutes-$date.txt";

  if (-f $link) {
      push(@vers,$link);
  }
  $link="$wgpath/$date/$sesnam-minutes-$date.txt";
  if (-f $link) {
      push(@vers,$link);
  }
  &urn_error("404 Not Found\n") if ($#vers==-1);

  print "Status: 200 OK\n";
  print "Content-type: multipart/alternative; boundary=endpart\n\n";
  foreach $i (@vers) {
      print "--endpart\n";
      if ($i =~ /html$/) {
          print "Content-Type: text/html\n\n";
      }
      if ($i =~ /txt$/) {
          print "Content-Type: text/plain\n\n";
      }
      if ($i =~ /ps$/) {
          print "Content-Type: application/postscript\n\n";
      }
      open(FILE, "$i");
      while (<FILE>) {
          print "$_";
      }
      close FILE;
  }
  print "--endpart\n";
}

sub resolve1 {
  my($flag,@bib,$i,$k,$j,$done,@ref);
```

```
  my($l,$link,@vers);
  my($scheme, $value) = @_;
  $scheme =~ tr/A-Z/a-z/;
  &urn_error("404 Not Found\n")if (!defined $pathbase{$scheme});
  my($try)="/ftp/$pathbase{$scheme}$value.txt";
  if (-f $try) {
      push(@vers, $try);
  }
  $try="/ftp/$pathbase{$scheme}$value.ps";
  if (-f $try) {
      push(@vers, $try);
  }
  $try="/ftp/$pathbase{$scheme}$value.html";
  if (-f $try) {
      push(@vers, $try);

  }
  print "Status: 200 OK\n";
  print "Content-type: multipart/alternative; boundary=endpart\n\n";
  foreach $i (@vers) {
      print "--endpart\n";
      if ($i =~ /html$/) {
          print "Content-Type: text/html\n\n";
      }
      if ($i =~ /txt$/) {
          print "Content-Type: text/plain\n\n";
      }
      if ($i =~ /ps$/) {
          print "Content-Type: application/postscript\n\n";
      }
      open(FILE, "$i");
      while (<FILE>) {
          print "$_";
      }
      close FILE;
  }
  print "--endpart\n";
}

sub resolveid {
  my($flag,@bib,$i,$k,$j,$done,@ref);
  my($l,$link,@vers);
  my($scheme) = "id";
  my($value) = @_;
  $scheme =~ tr/A-Z/a-z/;
  &urn_error("404 Not Found\n")if (!defined $pathbase{$scheme});
  my($try)="/ftp/$pathbase{$scheme}$value.txt";
  if (-f $try) {
```

```
      push(@vers, $try);
  }
  $try="/ftp/$pathbase{$scheme}$value.ps";
  if (-f $try) {
      push(@vers, $try);
  }
  $try="/ftp/$pathbase{$scheme}$value.html";
  if (-f $try) {
      push(@vers, $try);
  }
  print "Status: 200 OK\n";
  print "Content-type: multipart/alternative; boundary=endpart\n\n";
  foreach $i (@vers) {
      print "--endpart\n";
      if ($i =~ /html$/) {
          print "Content-Type: text/html\n\n";

      }
      if ($i =~ /txt$/) {
          print "Content-Type: text/plain\n\n";
      }
      if ($i =~ /ps$/) {
          print "Content-Type: application/postscript\n\n";
      }
      open(FILE, "$i");
      while (<FILE>) {
          print "$_";
      }
      close FILE;
  }
  print "--endpart\n";
}
sub urn_error {
  my($code) = @_; #store failure code here...

  print "Status: $code";
  print "Content-type: text/html\n\n<HTML>\n";
  print "<head><title>URN Resolution: I2Rs $code</title></head>\n";
  print "<BODY>\n";
  print "<h1>URN to URL resolution failed for the URN:</h1>\n";
  print "<hr><h3>$urn</h3>\n";
  print "</body>\n";
  print "</html>\n";
  exit;
}
```

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.