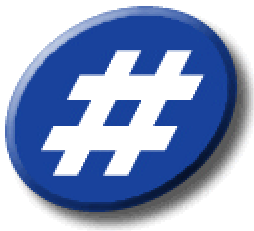
NetNumber ENUM Service
Programmer's Guide
For C

Version 2.0



NetNumber

TURNING TELEPHONE NUMBERS INTO INTERNET NUMBERS

Contents

Preface ..... **iii**
Who Should Read This Guide ..... **iii**
Where to Find NetNumber ENUM Service Information ..... **iii**
What's New in This Release ..... **iii**
What's In This Guide ..... **iv**
Document Conventions ..... **iv**
About the Sample Code ..... **iv**
Chapter 1 - Understanding the NetNumber ENUM Service **2**
NetNumber ENUM Service Overview 3
Tier-1 ENUM Service 5
Tier-2 ENUM Service 5
Chapter 2 – Setting Up the ENUM C Resolver API for Unix and Linux **6**
Minimum Requirements 6
Obtaining the API 6
Building the API 7
Makefile Example 9
Compiling the Client Applications 10
Example Build Output 10
Chapter 3 - Setting Up the ENUM C Resolver API for Windows NT / 2000 ..... **11**
Chapter 4 – Quick Start Example **12**
Understanding the Example Client Code 12
Example naptr_client.c 13
Example enum_client.c 13
Example enum_client.c 14
Running Example Clients 15
Chapter 5 – Querying the NetNumber ENUM Service **18**
Setting up a Unix Environment 18
NAPTR Functions 19
NAPTR Function Example 21
ENUM Functions 22
ENUM Function Example 23
Chapter – 6 Porting The C Resolver API **24**
Obtaining BIND 8 24

Preface

The NetNumber ENUM Service C Programmer's Guide documents the NetNumber ENUM Resolver C API, a development kit for querying NetNumber's ENUM directory service.
The chapter has the following sections:

• "Who Should Read This Guide"

• "Where to Find NetNumber ENUM Service Information"

"What's New in This Release"

• "What's in This Guide"

• "Where to Find Reference Information"

• "Documentation Conventions"

• "About the Sample Code"

Who Should Read This Guide

This guide is intended for use by C programmers who want to integrate NetNumber ENUM Service querying capabilities into IP telephony or messaging applications. It assumes that you are familiar with writing and building C code and the basic concepts of DNS. This manual is not intended for companies who wish to be a NetNumber ENUM Service Registrar. For more information on NetNumber ENUM Service Registrars please review our white paper at www.netnumber.com .

Where to Find NetNumber ENUM Service Information

For more information on the NetNumber ENUM Service visit our corporate web site at www.netnumber.com and download the technical white paper entitled "NetNumber ENUM Service"

What's New in This Release

Version 2 of the ENUM C Resolver API is more closely in line with the ENUM Java Resolver API. The functions within version 1 have been collapsed into 1 primary function for querying the NetNumber ENUM service. Version 2 provides the following functionality:

- Given a properly formed E.164 telephone number and service filter, return the associated set of NAPTR or ENUM records.
- Support for record sorting and NAPTR regular expression handling.
- The ability to serially query one or more corporate, commercial and/or public ENUM domains. (Example e164.company.com, e164.com and e164.arpa)
- An interface for querying for raw NAPTR records, as well as, ENUM records.

What's In This Guide

This guide explains how to write DNS client C code to query the NetNumber ENUM DNS Service. This guide is divided up into 2 parts:

- Part 1 – "Introduction to the NetNumber ENUM Service and ENUM", explains the architecture of the NetNumber ENUM Service.

- Part 2 – "Writing a Client to Query the NetNumber ENUM Service", provides detailed examples of how to program write a client program using NetNumber's ENUM Resolver C library.

Document Conventions

This manual uses the following font conventions:

• the `monospace font` is used for sample code and code listings, API and language elements (such as function names and class names), filenames, pathnames, directory names, HTML tags, and any text that must be typed on the screen. (***`Monospace italic font`*** is used for placeholders embedded in code.)

• *Italic type* is used for book titles, emphasis, variables and placeholders, and words used in the literal sense.

• **Boldface type** is used for glossary terms.

About the Sample Code

The sample code used throughout this manual was tested using the GNU C compiler on Solaris 7, 8, and Linux 2.6 and the Sun C compiler on Solaris 7, 8.

Part 1 - Introduction to the NetNumber ENUM Service and ENUM

Chapter 1 Understanding the NetNumber ENUM Service

This chapter provides an overview of the NetNumber ENUM Service architecture.

Chapter 2 Setting up the NetNumber ENUM Service Resolver C API for Unix

Detailed instructions on how to setup your Unix and Linux environment to begin using the ENUM C Resolver library and the examples in this guide.

Chapter 3 Setting up the NetNumber ENUM Service Resolver C API for Windows

Detailed instructions on how to setup your Windows NT 4.0 and 2000 environment to begin using the ENUM C Resolver library and the examples in this guide.

Chapter 4 Quick Start Example

A full working C program example on how to query the NetNumber ENUM Service and parse the directory's response.

Chapter 1 - Understanding the NetNumber ENUM Service

This chapter explains the architecture of the NetNumber ENUM Service. The NetNumber ENUM Service is a DNS based Internet directory service that resolves E.164 telephone numbers into one or moreInternet addresses.

If you have already read the white paper on the NetNumber ENUM Service located on the NetNumber website you may skip this chapter and go right to "Chapter 2 – Setting Up the NetNumber ENUM Service Client API".

This chapter is organized into the following sections:

- "NetNumber ENUM Service Overview"
- "A NetNumber ENUM Service Registrar"
- "Tier-1 ENUM Service"
- "Tier-2 ENUM Service"

NetNumber ENUM Service Overview

The NetNumber ENUM Service is a highly available, distributed Internet directory service that enables Internet-Telephony applications to translate standard E.164 telephone numbers into the Internet address information required to support a variety of IP-Telephony applications. For example:

- A SIP [8] User Agent or Proxy Server can query the NetNumber ENUM Service to locate a SIP address for setting up a real-time voice or fax call over an IP network like the Internet, a corporate intranet, or a managed extranet.
- An IP-enabled voicemail system can query the NetNumber ENUM Service to locate a VPIM address for sending a voicemail message over the Internet.
- A Unified-Messaging application can query the NetNumber ENUM Service to locate an SMTP address for sending a "spoken e-mail" message over the Internet.

Based on standard Internet technologies, the NetNumber ENUM Service architecture is simple, powerful, and extremely flexible. NetNumber deploys a logically two-tiered ENUM architecture [10], where the first tier is called the *Tier-1 ENUM Service or Registry* and the second tier is called the *Tier-2 ENUM Service or Registrar.*

The Tier-1 ENUM Service takes as input a properly formed E.164 telephone number and returns the location[1] of one or more second tier Tier-2 ENUM DNS name servers. The Tier-1 ENUM Service has been designed to operate as a top-level domain in the Internet Domain Name System (DNS) [1] and provides the following supportive services; DNS delegation, real-time provisioning, Registrant directory services, and a conflict resolution system.

The Tier-2 ENUM Service is responsible for providing authoritative Internet address information to the requesting telephony application. Implemented using DNS and NAPTR records, a Tier-2 ENUM name server provides an efficient access to a range of attribute information. The Tier-2 ENUM Service may be widely distributed across many different Tier-2 ENUM Service operators. NetNumber provides a fully out-sourced, 24x7 Tier-2 ENUM Service with support for real-time provisioning, E.164 number validation and Registrant directory services.

The two-tiered architecture provides for a highly scalable, distributed system that delivers a simple, ubiquitous approach for locating authoritative Internet addressing information associated with E.164 telephone numbers.

Figure 1 illustrates the two-tiered logical architecture of the NetNumber ENUM Service.

[1] The Tier-1 returns NS and/or A resource records (see RFC1034 and RFC2052).

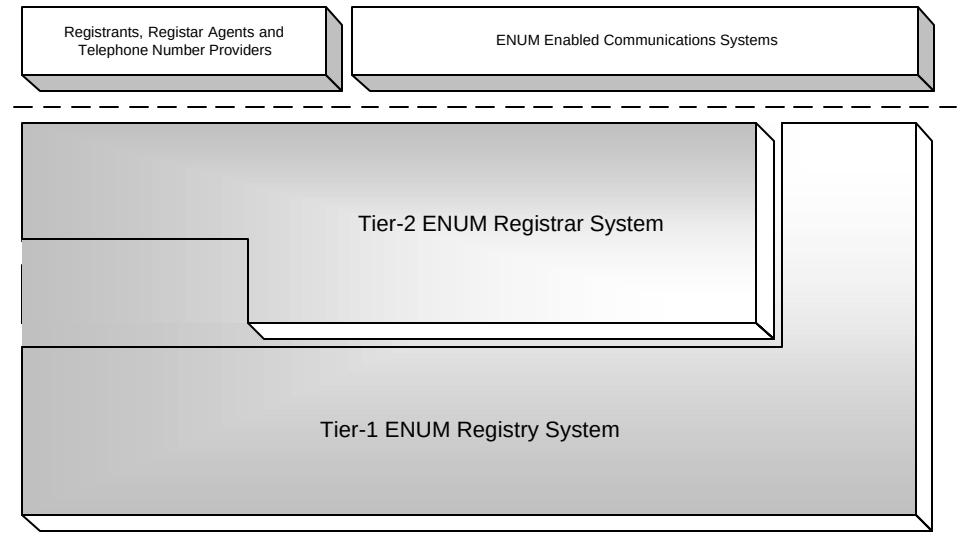


Figure 1 - ENUM Service Two-Tiered Architecture

Tier-1 ENUM Service

The Tier-1 ENUM Service is globally responsible for locating authoritative DNS servers containing NAPTR records associated with registered E.164 telephone numbers. The Internet community has standardized on DNS to provide the service of translating domain names into IP service and address information. The existing top-level domains of the Internet (com, net, org, edu, gov, etc.) are currently implemented within a well-defined regulatory and operational structure and the Tier-1 ENUM Service has been implemented in a fashion that is consistent with these top-level domains. The Tier-1 ENUM Service has been implemented within the “e164.com” sub-domain. Due to the large number of potential E.164 telephone numbers in the E.164 domain name space, the NetNumber ENUM Service has been physically partitioned across multiple DNS name server platforms.

Tier-2 ENUM Service

The Tier-2 ENUM Service is a federated registry of DNS ENUM compliant servers utilizing NAPTR records to provide authoritative Internet address and associated information in support of a variety of Internet-Telephony applications. An Tier-2 ENUM Service server contains many NAPTR records, which must be properly registered with the top-tier Tier-1 ENUM Service registry. Tier-2 ENUM Service directory servers do not require client authentication and may anonymously perform read-only operations. Anonymous add and modify operations are strictly prohibited.

Chapter 2 – Setting Up the ENUM C Resolver API for Unix and Linux

This chapter focuses on obtaining the ENUM Resolver C API for Unix and Linux users from the www.netnumber.com web site, installing it and setting it up. As of the date of this publication, this code has been tested on a Sun Solaris 2.7, 2.8 and Red Hat Linux 6.2 platform. As part of the API package a full version of the source code is provided so that this API may be easily ported to other platforms (see Chapter 6 for more details).

Minimum Requirements

Machine Requirements:

- Sun Sparc or PC with Pentium-class processor 90Mhz or higher
- 64 MB of RAM
- 2 MB of disk space for Zip package; 80 MB if the full BIND 8 source code package in installed.

Resolver Requirements:

- BIND v8.2.2 and above - libresolv.a or .so
- Unix – current binary is for Solaris 2.7/2.8 or Red Hat Linux 6.2. The zip package contains all of the source code to port it to other Unix platforms.
- gcc or comparable ANSII C compiler
- ar achiever

Obtaining the API

With a valid NetNumber login ID and password, log into www.netnumber.com. (For information on obtaining a registrant account go to www.netnumber.com). Once you have logged on, click on the Support tab and download link. Choose the C Resolver API file from the list and save it to your local machine.

The zipped file will contain the following files:

- enum_resolver.c – main library of ENUM resolver functions
- naptr_resolver.c – main library of NAPTR resolver functions.
- formatter.c – library of functions to format valid ENUM domain names from E.164 telephone numbers.
- enum_client.c – an example program that illustrates how to write an ENUM DNS client.
- naptr_client.c – an example program that illustrates how to write an NAPTR DNS client.

- resolvers.h – main header file containing function definitions for ENUM and NAPTR resolver functions.
- formatter.h – header files containing definitions for formatter library functions.
- makefile – a makefile for building the library under a Unix platform.

Building the API

After exploding the zip file, place the source files and include files in an appropriate directory, typically in a /src and /include subdirectory.

The following library files are required to build this API:

- `libresolv.a or libresolv.so –` basic DNS client resolver routines

Edit the makefile and make the following modifications:

- Set the CC local variable to the compiler you are using, either cc or gcc.
- Set the variable `ENUM_RESOLVER_DIR` to the main directory path where the source and include files are kept.

Once the makefile is setup to reflect your environment, issue the make command to build the libenum.a library file and the enum_client and naptr_client test programs. The following page contains the sample makefile included with the API package.

Makefile Example

```
####################
## Variables
####################
CC = gcc
#CC = cc
CFLAGS = -Wall -g

##
## Directories
##  Set RESOLVER_DIR to your appropriate base directory.
##
RESOLVER_DIR         = /nndev/resolver/C

RESOLVER_SRC_DIR     = $(RESOLVER_DIR)/src
RESOLVER_INC_DIR     = $(RESOLVER_DIR)/include
RESOLVER_LIB_DIR     = $(RESOLVER_DIR)/lib
RESOLVER_BIN_DIR     = $(RESOLVER_DIR)/bin

RESOLVER_LIB   = libenum.a


####################
## Top-Level Targets
####################

all: formatter naptr_resolver resolver naptr_client enum_client

clean:
       rm -f $(RESOLVER_BIN_DIR)/naptr_client \
              $(RESOLVER_BIN_DIR)/enum_client \
              $(RESOLVER_LIB_DIR)/*.[ao]

formatter:
       $(CC) $(CFLAGS) -I$(RESOLVER_INC_DIR) -c -o $(RESOLVER_LIB_DIR)/$@.o
$(RESOLVER_SRC_DIR)/$@.c
       ar -r $(RESOLVER_LIB_DIR)/$(RESOLVER_LIB) $(RESOLVER_LIB_DIR)/$@.o
       rm -f $(RESOLVER_LIB_DIR)/$@.o

naptr_resolver:
       $(CC) $(CFLAGS) -I$(RESOLVER_INC_DIR) -c -o $(RESOLVER_LIB_DIR)/$@.o
$(RESOLVER_SRC_DIR)/$@.c
       ar -r $(RESOLVER_LIB_DIR)/$(RESOLVER_LIB) $(RESOLVER_LIB_DIR)/$@.o
       rm -f $(RESOLVER_LIB_DIR)/$@.o

enum_resolver:
       $(CC) $(CFLAGS) -I$(RESOLVER_INC_DIR) -c -o $(RESOLVER_LIB_DIR)/$@.o
$(RESOLVER_SRC_DIR)/$@.c
       ar -r $(RESOLVER_LIB_DIR)/$(RESOLVER_LIB) $(RESOLVER_LIB_DIR)/$@.o
       rm -f $(RESOLVER_LIB_DIR)/$@.o

naptr_client:
       $(CC) $(CFLAGS) -L$(RESOLVER_LIB_DIR) -I$(RESOLVER_INC_DIR) -o
$(RESOLVER_BIN_DIR)/$@ $(RESOLVER_SRC_DIR)/$@.c -lresolv -lENUM

enum_client:
       $(CC) $(CFLAGS) -L$(RESOLVER_LIB_DIR) -I$(RESOLVER_INC_DIR) -o
$(RESOLVER_BIN_DIR)/$@ $(RESOLVER_SRC_DIR)/$@.c -lresolv -lENUM
```

Compiling the Client Applications

If you refer to the example makefile presented on the previous page there is an entry for compiling the C programs enum_resolver.c and naptr_resolver.c. These are client programs that utilize the libenum.a library. The following example is taken from the makefile to demonstrate how to build an application using the ENUM library.

```
gcc $(CFLAGS) -L$(ENUM_RESOLVER_LIB_DIR) -I$(ENUM_RESOLVER_INC_DIR) -o
$(ENUM_RESOLVER_BIN_DIR)/$@ $(ENUM_RESOLVER_SRC_DIR)/$@.c -lresolv -lENUM
```

Example Build Output

The following is the expected output from building the libenum.a, naptr_client, and enum_client executables:

```
enum_admin: make -f Makefile.solaris

gcc -Wall -g -I/export/home/test/enumres_c2.0/include -c -o /export/home/test/en
umres_c2.0/lib/formatter.o /export/home/test/enumres_c2.0/src/formatter.c
/usr/ccs/bin/ar -r /export/home/test/enumres_c2.0/lib/libenum.a /export/home/tes
t/enumres_c2.0/lib/formatter.o
rm -f /export/home/test/enumres_c2.0/lib/formatter.o
gcc -Wall -g -I/export/home/test/enumres_c2.0/include -c -o /export/home/test/en
umres_c2.0/lib/naptr_resolver.o /export/home/test/enumres_c2.0/src/naptr_resolve
r.c
/usr/ccs/bin/ar -r /export/home/test/enumres_c2.0/lib/libenum.a /export/home/tes
t/enumres_c2.0/lib/naptr_resolver.o
rm -f /export/home/test/enumres_c2.0/lib/naptr_resolver.o
gcc -Wall -g -I/export/home/test/enumres_c2.0/include -c -o /export/home/test/en
umres_c2.0/lib/enum_resolver.o /export/home/test/enumres_c2.0/src/enum_resolver.
c
/usr/ccs/bin/ar -r /export/home/test/enumres_c2.0/lib/libenum.a /export/home/tes
t/enumres_c2.0/lib/enum_resolver.o
rm -f /export/home/test/enumres_c2.0/lib/enum_resolver.o
gcc -Wall -g -L/export/home/test/enumres_c2.0/lib -I/export/home/test/enumres_c2
.0/include -o /export/home/test/enumres_c2.0/bin/naptr_client /export/home/test/
enumres_c2.0/src/naptr_client.c -lresolv -lENUM
gcc -Wall -g -L/export/home/test/enumres_c2.0/lib -I/export/home/test/enumres_c2
.0/include -o /export/home/test/enumres_c2.0/bin/enum_client /export/home/test/e
numres_c2.0/src/enum_client.c -lresolv -lENUM
```

Chapter 3 - Setting Up the ENUM C Resolver API for Windows NT / 2000

A release that supports the Microsoft Windows NT and 2000 operating system will be available at a later date. Please check back at our website www.netnumber.com .

Chapter 4 – Quick Start Example

Understanding the Example Client Code

The example code used as an illustration as to how a client program would query the NetNumber ENUM Service for an Internet address given an E.164 telephone number is the enum_client.c or naptr_client.c program included with the API package. The enum_client.c program does the following:

1. Resolves the E.164 domain name to a set of Internet addresses.
2. Frees the memory.

The following pages present the example code for enum_client.c and naptr_client.c .

Example naptr_client.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "formatter.h"
#include "resolvers.h"

#define MAX_RESULTS 20

int main(int argc, char **argv) {
   NAPTR_record *results[MAX_RESULTS];
   char **root_domains=NULL;
   char *client_default_domains[]={"trial.e164.com.",NULL,NULL,NULL};

   int i;
   int ret;

   /*
   ** Check command-line
   */
   if(argc<2) {
      fprintf(stderr,"usage: %s <E.164> [root domains (up to 4)] \n",argv[0]);
      exit(1);
   }
   root_domains=(argc>2) ? argv+2 : client_default_domains;

   /*
   ** Do the queries
   */
   ret = naptr_resolve(argv[1], root_domains, results, MAX_RESULTS);

   /*
   ** Display results
   */
   for(i=0;i<ret;i++) {
      printf("Result #%d\n domain = %s\n ttl = %lu\n order = %u\n pref = %u\n
flags = %s\n service = %s\n regexp = %s\n replacement = %s\n\n",
             i+1,
             results[i]->domain,
             results[i]->ttl,
             results[i]->order,
             results[i]->preference,
             results[i]->flags,
             results[i]->service,
             results[i]->regexp,
             results[i]->replacement);
       free_naptr_record(results[i]);
   }

   printf("Total number of results:%d\n",ret);

   exit(0);
```

Example enum_client.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "formatter.h"
#include "resolvers.h"

#define MAX_RESULTS 20

int main(int argc, char **argv) {
   ENUM_record *results[MAX_RESULTS];
   char **root_domains=NULL;
   char *client_default_domains[]={"trial.e164.com.",NULL,NULL,NULL};
   char *filter=NULL;
   int i;
   int ret;

   /*
   ** Check command-line
   */
   if(argc<3) {
      fprintf(stderr,"usage: %s <E.164> <filter|\"all\"> [root domains
(up to 4)] \n",argv[0]);
      exit(1);
   }
   if(strcmp(*(argv+2),"all")!=0 && strcmp(*(argv+2),"ALL")!=0)
     filter=*(argv+2);
   root_domains=(argc>3) ? argv+3 : client_default_domains;

   /*
   ** Do the queries
   */
   ret = enum_resolve(argv[1], filter, root_domains, results,
MAX_RESULTS);

   /*
   ** Display results
   */
   for(i=0;i<ret;i++) {
      printf("Result #%d\n domain = %s\n order = %u\n pref = %u\n service
= %s\n uri = %s\n\n",
             (i+1),
             results[i]->domain,
             results[i]->order,
             results[i]->preference,
             results[i]->service,
             results[i]->uri);
      free_enum_record(results[i]);
   }

   printf("Total number of results:%d\n",ret);
   exit(0);
}
```

Running Example Clients

Before running either of the example client programs, make sure that the machine your are running on is configured to use a DNS server that can recurse out to the Internet.

naptr_client example:

```
enum_admin: ./naptr_client 11112220000
Result #1
 domain = 0.0.0.0.2.2.2.1.1.1.1.e164.com
 ttl = 60
 order = 0
 pref = 0
 flags = u
 service = SIP+E2U
 regexp = !^.*$!SIP:38.136.73.87!
 replacement = .

Result #2
 domain = 0.0.0.0.2.2.2.1.1.1.1.e164.com
 ttl = 60
 order = 1
 pref = 1
 flags = u
 service = SMTP+E2U
 regexp = !^.*$!mailto:jsmith@company.com!
 replacement = .

Result #3
 domain = 0.0.0.0.2.2.2.1.1.1.1.e164.com
 ttl = 60
 order = 2
 pref = 1
 flags = u
 service = VPIM+E2U
 regexp = !^.*$!mailto:jsmith@vmail.company.com!
 replacement = .

Result #4
 domain = 0.0.0.0.2.2.2.1.1.1.1.e164.com
 ttl = 60
 order = 3
 pref = 1
 flags = u
 service = H323+E2U
 regexp = !^.*$!h323:jsmith@voice.company.com!
 replacement = .

Total number of results:4
```

enum_client example:

```
enum_admin: ./enum_client 11112220000 all
Result #1
 domain = 0.0.0.0.2.2.2.1.1.1.1.e164.com
 order = 0
 pref = 0
 service = SIP
 uri = SIP:38.136.73.87

Result #2
 domain = 0.0.0.0.2.2.2.1.1.1.1.e164.com
 order = 1
 pref = 1
 service = SMTP
 uri = mailto:jsmith@company.com

Result #3
 domain = 0.0.0.0.2.2.2.1.1.1.1.e164.com
 order = 2
 pref = 1
 service = VPIM
 uri = mailto:jsmith@vmail.company.com

Result #4
 domain = 0.0.0.0.2.2.2.1.1.1.1.e164.com
 order = 3
 pref = 1
 service = H323
 uri = h323:jsmith@voice.company.com

Total number of results:4
```

Part 2 - Writing Client code to Query the NetNumber ENUM Service

Chapter 5 *Querying the NetNumber ENUM Service*
This chapter provides additional information for utilizing the C ENUM Resolver API to query the NetNumber ENUM Service for Internet addresses.

Chapter 6 *Porting the C Resolver API*
This chapter provides basic information on porting the Resolver library.

Chapter 5 – Querying the NetNumber ENUM Service

This chapter expands on the C API functions that were discussed as part of the naptr_client.c and enum_client.c examples. .The API now has 2 families of functions, one that queries for raw NAPTR records and another to query for ENUM records which encapsulate the NAPTR information into a defined structure. This chapter will cover both the NAPTR and ENUM functions.

The following sections are contained in this chapter:

- “Setting up a Unix Environment”
- “Setting up a Windows Environment”
- NAPTR Functions
- ENUM Functions

Setting up a Unix Environment

The Unix examples in this guide were compiled and tested in a Sun Solaris 2.7 and 2.8 environment using the gcc compiler.

The following libraries and include files are used:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "formatter.h"
#include "resolvers.h"
```

Libraries to link in:

```
/lib/libresolv.a
```

Note: If your system does not include libresolv.a or does not include all of the ns_ functions contained in the BIND 8.2.x release please see Chapter 6 “Porting the C Resolver API”.

See Chapter 3 or 4 for information on setting up makefiles and building your code using the API.

NAPTR Functions

At the core of ENUM resolution is the NAPTR record that is returned from a query on a E.164 number. These functions return only the raw NAPTR records and do not provide any regular expression processing.

NAPTR Query

Querying for NAPTR records returns an NAPTR_record structure containing ALL of the associated NAPTR records associated with that E.164 name.

The structure is defined as:

```
typedef struct naptr_rec {
   char *domain;
   unsigned long ttl;
   unsigned int order;
   unsigned int preference;
   char *flags;
   char *service;
   char *regexp;
   char *replacement;
} NAPTR_record;
```

Functions

```
int naptr_resolve(char *e164_number,
                                char **root_domains,
                                NAPTR_record **results,
                                int max_results)
```

Description

Retrieves NAPTR records associated with a given E.164 number on given root domain. Multiple root domains can be queried by providing an array of root domains. .

Parameters

char *e164_number : null terminated E.164 number string (telephone #)
char **root domains : pointer to an array of root domain strings.
Setting this to NULL will default to "e164.com"

NAPTR_record **results : pre-allocated buffer for results.
int max_results : size of results buffer (in number of NAPTR_records)

Returns

Returns the total number of NAPTR records returned. If the query does not return any NAPTR records the function returns 0.

```
void free_naptr_record(NAPTR_record *naptr_record)
```

Description

Releases all underlying memory allocations for a NAPTR_record.

Parameters

NAPTR_record *naptr_record: name of the NAPTR structure

NAPTR Function Example

The following example illustrates the use of the NAPTR functions.

```
NAPTR_record *results[MAX_RESULTS];
char *e164 = “18005551212”;
char **root_domains={“trial.e164.com.”,NULL};
int i;
int ret;

ret=naptr_resolve(      e164,
                              root_domains,
                              results,
                              MAX_RESULTS);

for(i=0;i<ret;i++) {
      free_naptr_record(results[i]);
}
```

ENUM Functions

The NetNumber ENUM C Resolver API provides a set of functions that procesws the raw NAPTR record data into order, preference, service type and URL fields for each NAPTR record returned. It also sorts the records based on Order and then Preference specified in the NAPTR record fields.

ENUM Query

Querying for ENUM records returns an ENUM_record structure for a specified service type filter or all of ENUM _records.

The structure is defined as:

```
typedef struct enum_rec {
   char *domain;
   unsigned int order;
   unsigned int preference;
   char *service;
   char *uri;
} ENUM_record;
```

Functions

```
int enum_resolve(char *e164_number,
                        char *service_filter,
                        char **root_domains,
                        ENUM_record **results,
                        int max_results)
```

Description

Retrieves NAPTR records associated with a given E.164 number on given root domain and process into ENUM records. Multiple root domains can be queried by providing an array of root domains. The root domains are queried serially in the order they appear in the array.

Parameters

char *e164_number : null terminated E.164 number string (telephone #)

char *service_filter : string of the specific service type you are looking for (ex. SIP, H323, TEL or ALL).
char **root domains : pointer to an array of root domain strings. Setting this to NULL will default to "e164.com" and "e164.arpa".

ENUM_record **results : pre-allocated buffer for results.
int max_results : size of results buffer (in number of NAPTR_records)

Returns

Returns the total number of ENUM records returned for a given service type filter or all if all is specified. If there are no entries for an E.164 name, the function will return 0.

`void free_enum_record(ENUM_record *enum_record)`

Description

Releases all underlying memory allocations for a ENUM_record.

Parameters

ENUM_record: name of the ENUM structure

ENUM Function Example

The following illustrates the use the ENUM functions:

```
ENUM_record *results[MAX_RESULTS];
char *e164=”18005551212”;
char **root_domains=”trial.e164.com”;
char *filter=NULL;
int i;
int ret;

ret = enum_resolve(e164,
                           filter,
                           root_domains,
                           results,
                           MAX_RESULTS);

for(i=0;i<ret;i++) {
      free_enum_record(results[i]);
}
```

Chapter – 6 Porting The C Resolver API

The ENUM Resolver C API has been written to utilize the BIND DNS libraries included with most operating systems. There are however instances where that support does not exist or at an older BIND version that does not support NAPTR records. For instance the DNS resolver libraries that ship with Solaris 7 or 8 libresolv.a are sufficient to build the ENUM resolver library and perform ENUM queries, whereas, Windows NT and 2000 do not include support for NAPTR records. The basic remedy for any system that does not contain basic support for BIND 8.2.x or later is to build a base BIND package.

Obtaining BIND 8

To download a copy of the BIND distribution go to the ISC website at:

http://www.isc.org/products/BIND/bind8.html

Please refer to the build instructions within the BIND distribution to build the libbind.a library.