**Multi-Threaded DNS64 server user documentation**

**Technical documentation**

This program is intended to be a fast multi-threaded application which can provide DNS64 server functions. The software is named MTD64 and accessible as a GPL v2.0 licensed open source code. For speed related reason most part of the source code is written in C but it contains a few C++ parts as well. The software package consists of three .cpp files, a header.h file, a configuration file named settings.conf and a Makefile, each files are well documented. This program uses no special libraries only the standard built-ins therefore it can be considered as a cross platform software. The program requires a compiler which supports the ISO/IEC 14882:2011 standards commonly known as C++11 (formerly named C++0x). Using the Makefile the MTD64 can be complied by running the “make” command. The Makefile script will check the version of the GCC compiler, if it is 4.7 or greater then the program going to be compiled with the -std=c++11 flag otherwise ‑std=c++0x flag going to be used. The server program is a multi-threaded application therefore higher performance can be reached if the architecture in which the program will be run has multi-core support. After a successful compiling the executable will be placed into the working directory named as “mtd64”. Files created by the compiling process can be removed using the “make clean” command.

Running the MTD64 server requires administrator privileges since a port listening will be established. In addition the system firewall should be check whether the incoming UDP packets are accepted on port 53.  
The server also requires an IPv6 and an IPv4 interface which can be the same interface. Since the MTD64 reaches the recursive DNS servers via IPv4 so unless there is no DNS server within the local network it is necessary to have Internet connection in order to remote servers can be used for domain name resolution. There is no restriction regarding link-local/site-local IPv6 addresses both can be used equally. The server program performs logging operation which uses the services provided by the syslog. The messages will be written into syslog as DNS64SERVER into the LOG\_DAEMON facility.

**Configuration Documentation**

In order to function as expected it is necessary to have proper configuration settings regarding the MTD64. The file named settings.conf contains several options which can determine the operation of the program. The settings will be loaded by the executable mtd64 once when we start the program. The program reads the file line by line and will not interpret those line which start with “#” or “//”. In the configuration file before and after the commands there could be arbitrary number of spaces and tabs if the line length is no longer than 254 characters. These are the available eight commands which can be used as a setting within the settings.conf configuration file:

**nameserver:**

With the nameserver command followed by a valid IPv4 address we can add DNS nameservers which going to be used for resolving a domain name for example: “nameserver 8.8.4.4”. It is possible to use the same nameservers what the OS is using. For this the “defaults” parameters should be added for example: “nameserver defaults”. In case there are multiple “nameserver” settings in the configuration file DNS servers going to be added together up to 50 which is the maximum number what program supports. If there is no “nameserver” setting in the configuration file MTD64 going to load the nameservers used by OS.

**selection-mode:**

There are two selection algorithms available for choosing between the given nameservers for domain name resolution. The first is the random mode which will select a random DNS server in every case when MTD64 has to send out a query. The second algorithm called round-robin will use the same DNS server till it is responding to the sent queries within time. MTD64 will only select a next server if it does not respond once within the time. The algorithms can be set with “selection-mode random” or “selection-mode round-robin”. If there is no such setting in the configuration file the default will be set which is round-robin.

**timeout-time-sec:**

With this command followed by a not negative number can be set the packet arrival expectation time in seconds before the MTD64 will try to resend the packets (if the resending is enabled). For example: “timeout-time-sec 2”.

**timeout-time-usec:**

With this command followed by a not negative number can be set the packet arrival expectation time in micro seconds before the MTD64 will try to resend the packets (if the resending is enabled). If both the timeout-time-sec and timeout-time-usec settings are missing then 1.25 seconds will be set as the arrival expectation time.

**resend-attempts:**

By this command we can tell the MTD64 how many times it have to try to resend a packet which has not been answered within the given expectation time. For example: “resend-attempts 2”

**dns64-prefix:**

This important setting will determine what IPv4-Embedded IPv6 address should be synthesized in case of there is no “AAAA” record. It is mandatory to specify a valid IPv6 address linked with a proper prefix with a slash sign. For example: 2001:db8::/64. Program only accepts prefixes which are specified in the 2.2 chapter in RFC 6052 which are: 32, 40, 48, 56, 64, and 96. In case of no dns64-prefix setting can be found in the configuration file the NAT64 Well-known Prefix is going to be set which is 64:ff9b::/96.

**response-maxlength:**

With the response-maxlenght command followed by a not negative number can be the maximum length value of the IPv6 response message set. The reason why this option is useful is that there may be cases when the “A” response message is almost 512 byte which is the maximum size of a DNS message according to RFC 5966. If such DNS message is going to be synthesized as “AAAA” the length of the DNS message may exceed the limit due to the fact containing an IPv6 address instead of IPv4 requires 12 additional bytes. By setting the limit to 512 with using the setting “response-maxlenght 512” MTD64 is not going to send DNS message longer than 512 byte it will cut of the last entry which not fitted in (the consistency of DNS message will remain). Since there are programs which can accept oversized DNS messages this option can be useful because we can set the maximum value greater than 512.

**debugging:**

By the “debugging yes” setting we can enable verbose logging into the syslog. By default this is not enabled.