DTS Application Library

0.00

Generated by Doxygen 1.8.2

Sun Oct 13 2013 07:45:21

Contents

Chapter 1

Todo List

Global daemonize ()

WIN32 options is there a alternative for this.

Global framework_mkcore (char *progname, char *name, char *email, char *web, int year, char *runfile, int flags, syssighandler sigfunc)

does threads actually work in windows with no sighandler.

Global seedrand (void)

This wont work on WIN32

Global touch (const char *filename, uid_t user, gid_t group)

WIN32 does not use uid/gid and move to file utils module.

Global zuncompress (struct zobj *buff, uint8_t *obuff)

Implement this without needing original buff len using inflate

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Module Documentation

5.1 Distrotech Application Library

A Collection of helper functions and wrapped up interfaces to other libraries.

Modules

- INI Style config file Interface

  Reads a ini config file into grouped hashed buckets.
- CURL Url interface.

  Interface to libCURL.
- File utility functions

  Convinece wrappers arround stat.
- Linux network interface functions

  Implement various interface routines from libnetlink.
- IPv4 and IPv6 functions

  Helper functions for various calculations.
- XML Interface

  Utilities for managing XML documents.
- XSLT Interface

  Utilities for managing XML documents.
- Burtle Bob hash algorythim.

  Default init value for hash function.
- Referenced Objects

  Utilities for managing referenced objects.
- Posix thread interface

  Functions for starting and managing threads.
- Unix socket thread

  Attach a thread to a unix socket calling a callback on connect.
- Micelaneous utilities.

  Utilities commonly used.
- Zlib Interface

  Simplified interface to Compress/Uncompress/Test a buffer.

Files

- file dtsapp.h

  DTS Application library API Include file.
- file main.c

  Application framework.

Macros

- #define FRAMEWORK_MAIN(progname, name, email, www, year, runfile, flags, sighfunc)

  A macro to replace main() with initilization and daemonization code.

Typedefs

- typedef int(∗ frameworkfunc)(int, char ∗∗)

  Framework callback function.
- typedef void(∗ syssighandler)(int, siginfo_t ∗, void ∗)

  Callback to user supplied signal handler.

Enumerations

- enum framework_flags { FRAMEWORK_FLAG_DAEMON = 1 << 0, FRAMEWORK_FLAG_NOGNU = 1 << 1, FRAMEWORK_FLAG_NOTHREAD = 1 << 2 }

  Application control flags.

Functions

- void printgnu ()

  Print a brief GNU copyright notice on console.
- void daemonize ()

  Daemonise the application using fork/exit.
- int lockpidfile (const char ∗runfile)

  Lock the run file in the framework application info.
- void framework_mkcore (char ∗progname, char ∗name, char ∗email, char ∗web, int year, char ∗runfile, int flags, syssighandler sigfunc)

  Initilise application data structure and return a reference.
- int framework_init (int argc, char ∗argv[], frameworkfunc callback)

  Initilise the application daemonise and join the manager thread.

5.1.1 Detailed Description

A Collection of helper functions and wrapped up interfaces to other libraries.

5.1.2 Macro Definition Documentation

5.1.2.1 #define FRAMEWORK_MAIN(*progname,* **name**, *email, www, year, runfile, flags, sighfunc*)

Value:

```
static int  framework_main(int argc, char *argv[]); \
int  main(int argc, char *argv[]) { \
    framework_mkcore(progname, name, email, www, year, runfile, flags,
       sighfunc); \
    return (framework_init(argc, argv, framework_main)); \
} \
static int  framework_main(int argc, char *argv[])
```

A macro to replace main() with initilization and daemonization code.

See Also

framework_flags
framework_mkcore()
framework_init()

Parameters

progname	Descriptive program name.
name	Copyright holders name.
email	Copyright holders email.
www	Web address.
year	Copyright year.
runfile	Application runfile.
flags	Application flags.
sighfunc	Signal handler function.

Definition at line 667 of file dtsapp.h.

5.1.3 Typedef Documentation

5.1.3.1 typedef int(* frameworkfunc)(int, char **)

Framework callback function.

Parameters

argc	Argument count.
argv	Argument array.

Returns

Application exit code.

Definition at line 140 of file dtsapp.h.

5.1.3.2 typedef void(* syssighandler)(int, siginfo_t *, void *)

Callback to user supplied signal handler.

Parameters

sig	Signal been handled.
si	Sa sigaction.
unsed	Unused cast to void from ucontext_t

Definition at line 148 of file dtsapp.h.

5.1.4 Enumeration Type Documentation

5.1.4.1 enum framework_flags

Application control flags.

Enumerator:

FRAMEWORK_FLAG_DAEMON Allow application daemonization.

FRAMEWORK_FLAG_NOGNU Dont print GNU copyright.

FRAMEWORK_FLAG_NOTHREAD Dont start thread manager.

Definition at line 181 of file dtsapp.h.

```
                    {
    FRAMEWORK_FLAG_DAEMON   = 1 << 0,
    FRAMEWORK_FLAG_NOGNU    = 1 << 1,
    FRAMEWORK_FLAG_NOTHREAD  = 1 << 2
};
```

5.1.5 Function Documentation

5.1.5.1 void daemonize ( )

Daemonise the application using fork/exit.

This should be run early before file descriptors and threads are started

See Also

FRAMEWORK_MAIN()

Warning

on failure the program will exit.

Todo WIN32 options is there a alternative for this.

Definition at line 96 of file main.c.

Referenced by framework_init().

```
                        {
#ifndef __WIN32__
    pid_t   forkpid;

    /* fork and die daemonize*/
    forkpid = fork();
    if (forkpid > 0) {
        /* im all grown up and can pass onto child*/
        exit(0);
    } else if (forkpid < 0) {
        /* could not fork*/
        exit(-1);
    }

    /* Dont want these as a daemon*/
    signal(SIGTSTP, SIG_IGN);
    signal(SIGCHLD, SIG_IGN);
#endif
}
```

5.1.5.2 int framework_init (int *argc*, char * *argv[]*, frameworkfunc *callback*)

Initilise the application daemonise and join the manager thread.

Warning

failure to pass a callback will require running stopthreads and jointhrea..
framework information configured by framework_mkcore will be freed on exit.

Parameters

argc	Argument count argv[0] will be program name.
argv	Argument array.
callback	Function to pass control too.

Definition at line 246 of file main.c.

References daemonize(), framework_core::flags, framework_core::flock, FRAMEWORK_FLAG_DAEMON, FRAMEWORK_FLAG_NOGNU, FRAMEWORK_FLAG_NOTHREAD, jointhreads(), lockpidfile(), objunref(), printgnu(), framework_core::runfile, framework_core::sa, seedrand(), sslstartup(), startthreads(), and stopthreads().

```
                                                                    {
    int ret = 0;

    seedrand();
    sslstartup();

    /*prinit out a GNU licence summary*/
    if (!(framework_core_info->flags & FRAMEWORK_FLAG_NOGNU
      )) {
        printgnu();
    }

    /* fork the process to daemonize it*/
    if (framework_core_info->flags & FRAMEWORK_FLAG_DAEMON
      ) {
        daemonize();
    }

    if ((framework_core_info->flock = lockpidfile(
      framework_core_info->runfile) < 0)) {
        printf("Could not lock pid file Exiting\n");
        objunref(framework_core_info);
        return (-1);
    }

#ifndef __WIN32__
    /* interupt handler close clean on term so physical is reset*/
    configure_sigact(framework_core_info->sa);
#endif

    /*init the threadlist start thread manager*/
    if (!(framework_core_info->flags & FRAMEWORK_FLAG_NOTHREAD
      ) && !startthreads()) {
        printf("Memory Error could not start threads\n");
        objunref(framework_core_info);
        return (-1);
    }

    /*run the code from the application*/
    if (callback) {
        ret = callback(argc, argv);
        /* wait for all threads to end*/
        stopthreads();
        jointhreads();
    }

    /* turn off the lights*/
    objunref(framework_core_info);
    return (ret);
}
```

5.1.5.3 void framework_mkcore (char * *progname*, char * *name*, char * *email*, char * *web*, int *year*, char * *runfile*, int *flags*, syssighandler *sigfunc*)

Initilise application data structure and return a reference.

Note

The returned value must be un referenced

Warning

failure to supply a signal handler on non WIN32 systems will deafault to exiting with -1 on SIGINT/SIGKILL.

Todo does threads actually work in windows with no sighandler.

Warning

do not call this function without calling framework_init as the memory allocated will not be freed.

Parameters

progname	Descrioptive program name.
name	Copyright holder.
email	Copyright email address.
web	Website address.
year	Copyright year.
runfile	Run file that will store the pid and be locked (flock).
flags	Application flags.
sigfunc	Signal handler.

Definition at line 207 of file main.c.

References ALLOC_CONST, framework_core::developer, framework_core::email, framework_core::flags, malloc, objalloc(), objunref(), framework_core::progname, framework_core::runfile, framework_core::sa, framework_core::sig_handler, framework_core::www, and framework_core::year.

```
                                                                    {
    struct framework_core *core_info;
    if (framework_core_info) {
        objunref(framework_core_info);
        framework_core_info = NULL;
    }

    if (!(core_info = objalloc(sizeof(*core_info), framework_free))) {
        return;
    }

#ifndef __WIN32__
    if (core_info && !(core_info->sa = malloc(sizeof(*core_info->sa))
      )) {
        free(core_info);
        return;
    }
#endif

    ALLOC_CONST(core_info->developer, name);
    ALLOC_CONST(core_info->email, email);
    ALLOC_CONST(core_info->www, web);
    ALLOC_CONST(core_info->runfile, runfile);
    ALLOC_CONST(core_info->progname, progname);
    core_info->year = year;
    core_info->flags = flags;
#ifndef __WIN32__
    core_info->sig_handler = sigfunc;
#endif
    /* Pass reference to static system variable*/
    framework_core_info = core_info;
```

```
}
```

5.1.5.4 int lockpidfile (const char * *runfile*)

Lock the run file in the framework application info.

Parameters

runfile	File to write pid to and lock.

Returns

0 if no file is specified or not supported. The file descriptor on success.

Definition at line 120 of file main.c.

References framework_core::flock.

Referenced by framework_init().

```
                                                {
    int lck_fd = 0;
#ifndef __WIN32__
    char pidstr[12];
    pid_t   mypid;

    mypid = getpid();
    sprintf(pidstr,"%i\n", (int)mypid);
    if (runfile && ((lck_fd = open(runfile, O_RDWR|O_CREAT, 0640)
      ) > 0) && (!flock(lck_fd, LOCK_EX | LOCK_NB))) {
        if (write(lck_fd, pidstr, strlen(pidstr)) < 0) {
            close(lck_fd);
            lck_fd = -1;
        }
    /* file was opened and not locked*/
    } else if (runfile && lck_fd) {
        close(lck_fd);
        lck_fd = -1;
    }
#endif
    return (lck_fd);
}
```

5.1.5.5 void printgnu ( )

Print a brief GNU copyright notice on console.

framework_mkcore() needs to be run to ininitilise the data

See Also

FRAMEWORK_MAIN()
framework_mkcore()

Definition at line 75 of file main.c.

References framework_core::developer, framework_core::email, objref(), objunref(), framework_core::progname, framework_core::www, and framework_core::year.

Referenced by framework_init().

```
                        {
    struct framework_core *ci;
    if (framework_core_info && objref(framework_core_info)) {
        ci = framework_core_info;
    } else {
        return;
```

```
    }
    printf("%s\n\nCopyright (C) %i %s <%s>\n"
           "        %s\n\n"
           "    This program comes with ABSOLUTELY NO WARRANTY\n"
           "    This is free software, and you are welcome to redistribute it\n
      "
           "    under certain condition\n\n", ci->progname, ci->year
      , ci->developer, ci->email, ci->www);
    objunref(ci);
}
```

5.2 INI Style config file Interface

Reads a ini config file into grouped hashed buckets.

Data Structures

- struct config_category
- struct config_file

Functions

- void initconfigfiles (void)
- void unrefconfigfiles (void)
- int process_config (const char *configname, const char *configfile)
- struct bucket_list * get_config_file (const char *configname)
- struct bucket_list * get_config_category (const char *configname, const char *category)
- struct bucket_list * get_category_next (struct bucket_loop *cloop, char *name, int len)
- struct bucket_loop * get_category_loop (const char *configname)
- void config_entry_callback (struct bucket_list *entries, config_entrycb entry_cb)
- void config_cat_callback (struct bucket_list *categories, config_catcb cat_cb)
- void config_file_callback (config_filecb file_cb)
- struct config_entry * get_config_entry (struct bucket_list *categories, const char *item)

5.2.1 Detailed Description

Reads a ini config file into grouped hashed buckets.

5.2.2 Function Documentation

5.2.2.1 void config_cat_callback (struct bucket_list * *categories,* config_catcb *cat_cb*)

Definition at line 344 of file config.c.

References bucketlist_callback().

```
        {
    bucketlist_callback(categories, category_callback, &
      cat_cb);
}
```

5.2.2.2 void config_entry_callback (struct bucket_list * *entries,* config_entrycb *entry_cb*)

Definition at line 331 of file config.c.

References bucketlist_callback().

```
            {
    bucketlist_callback(entries, entry_callback, &entry_cb);
}
```

5.2.2.3 void config_file_callback (config_filecb *file_cb*)

Definition at line 357 of file config.c.

References bucketlist_callback().

```
                                                        {
    bucketlist_callback(configfiles, file_callback, &file_cb
      );
}
```

5.2.2.4 struct bucket_loop* get_category_loop (const char * *configname*) [read]

Definition at line 312 of file config.c.

References get_config_file(), init_bucket_loop(), and objunref().

```
                                                                    {
    struct bucket_loop *cloop;
    struct bucket_list *file;

    file = get_config_file(configname);
    cloop = init_bucket_loop(file);
    objunref(file);
    return (cloop);
}
```

5.2.2.5 struct bucket_list* get_category_next (struct bucket_loop * *cloop,* char * *name,* int *len*) [read]

Definition at line 291 of file config.c.

References config_category::entries, config_category::name, next_bucket_loop(), objref(), objunref(), and strlenzero().

```
                  {
    struct config_category *category;

    if (cloop && (category = next_bucket_loop(cloop))) {
        if (category->entries) {
            if (!objref(category->entries)) {
                objunref(category);
                return (NULL);
            }
            if (!strlenzero(name)) {
                strncpy(name, category->name, len);
            }
            objunref(category);
            return (category->entries);
        } else {
            objunref(category);
        }
    }
    return (NULL);
}
```

5.2.2.6 struct bucket_list* get_config_category (const char * *configname,* const char * *category*) [read]

Definition at line 267 of file config.c.

References bucket_list_find_key(), config_category::entries, get_config_file(), objref(), and objunref().

```
                  {
    struct bucket_list *file;
    struct config_category *cat;

    file = get_config_file(configname);
```

```
    if (category) {
        cat = bucket_list_find_key(file, category);
    } else {
        cat = bucket_list_find_key(file, "default");
    }

    objunref(file);
    if (cat) {
        if (!objref(cat->entries)) {
            objunref(cat);
            return (NULL);
        }
        objunref(cat);
        return (cat->entries);
    } else {
        return (NULL);
    }
}
```

5.2.2.7 struct config_entry* get_config_entry (struct bucket_list * *categories*, const char * *item*) [read]

Definition at line 361 of file config.c.

References bucket_list_find_key().

```
                    {
    struct config_entry *entry;

    entry = bucket_list_find_key(categories, item);

    return (entry);
}
```

5.2.2.8 struct bucket_list* get_config_file (const char * *configname*) [read]

Definition at line 250 of file config.c.

References bucket_list_find_key(), config_file::cat, objref(), and objunref().

Referenced by get_category_loop(), and get_config_category().

```
                                                                    {
    struct config_file *file;

    if ((file = bucket_list_find_key(configfiles,
      configname))) {
        if (file->cat) {
            if (!objref(file->cat)) {
                objunref(file);
                return (NULL);
            }
            objunref(file);
            return (file->cat);
        }
        objunref(file);
    }
    return (NULL);
}
```

5.2.2.9 void initconfigfiles (void)

Definition at line 66 of file config.c.

References create_bucketlist().

Referenced by process_config().

```
                                  {
    if (!configfiles) {
        configfiles = create_bucketlist(4, hash_files);
    }
}
```

5.2.2.10 int process_config (const char * *configname*, const char * *configfile*)

Definition at line 187 of file config.c.

References addtobucket(), config_file::cat, config_file::filepath, initconfigfiles(), objunref(), strlenzero(), and trim().

```
                                                                        {
    struct config_file *file;
    struct config_category *category = NULL;
    FILE *config;
    char line[256];
    char item[128];
    char value[128];
    char *tmp = (char *)&line;
    char *token;

    if (!configfiles) {
        initconfigfiles();
    }

    file = create_conf_file(configname, configfile);
    addtobucket(configfiles, file);

    if (!(config = fopen(file->filepath, "r"))) {
        return (-1);
    }

    while(fgets(line, sizeof(line) - 1, config)) {
        if (!(tmp = filterconf(line, 3))) {
            continue;
        }

        /*this is a new category*/
        if ((token = strchr(tmp, '[')) && (token == tmp)) {
            tmp++;
            token = strrchr(tmp, ']');
            token[0] = '\0';
            tmp = trim(tmp);
            if (!strlenzero(tmp)) {
                if (category) {
                    objunref(category);
                }
                category = create_conf_category(tmp);
                addtobucket(file->cat, category);
            }
            continue;
        }

        if (sscanf(tmp, "%[^=] %*[=] %[^\n]", (char *)&item, (char *)&value) !=
       2) {
            continue;
        }

        if (!category) {
            category = create_conf_category("default");
            addtobucket(file->cat, category);
        }

        add_conf_entry(category, trim(item), trim(value));
    }
    fclose(config);
    if (category) {
        objunref(category);
    }
    if (file) {
        objunref(file);
    }
    return (0);
}
```

5.2.2.11 void unrefconfigfiles (void)

Definition at line 72 of file config.c.

References objunref().

```
                                    {
    if (configfiles) {
        objunref(configfiles);
    }
}
```

5.3 CURL Url interface.

Interface to libCURL.

Data Structures

- struct **curl_progress**
- struct **curl_password**
- struct curl_post

Functions

- int curlinit (void)
- void curlclose (void)
- struct curlbuf * curl_geturl (const char *def_url, struct basic_auth *bauth, curl_authcb authcb, void *auth_data)
- struct curlbuf * curl_posturl (const char *def_url, struct basic_auth *bauth, struct curl_post *post, curl_authcb authcb, void *auth_data)
- struct curlbuf * curl_ungzip (struct curlbuf *cbuf)
- struct basic_auth * curl_newauth (const char *user, const char *passwd)
- void free_post (void *data)
- struct curl_post * curl_newpost (void)
- void curl_postitem (struct curl_post *post, const char *name, const char *item)
- char * url_escape (char *url)
- char * url_unescape (char *url)
- void free_progress (void *data)
- void curl_setprogress (curl_progress_func cb, curl_progress_pause p_cb, curl_progress_newdata d_cb, void *data)
- void free_curlpassword (void *data)
- void curl_setauth_cb (curl_authcb auth_cb, void *data)
- struct xml_doc * curl_buf2xml (struct curlbuf *cbuf)

5.3.1 Detailed Description

Interface to libCURL.

5.3.2 Function Documentation

5.3.2.1 struct xml_doc* curl_buf2xml (struct curlbuf * *cbuf*) [read]

Definition at line 413 of file curl.c.

References curlbuf::body, curlbuf::bsize, curlbuf::c_type, curl_ungzip(), and xml_loadbuf().

```
                                                        {
    struct xml_doc *xmldoc = NULL;

    if (cbuf && cbuf->c_type && !strcmp("application/xml", cbuf->c_type
      )) {
        curl_ungzip(cbuf);
        xmldoc = xml_loadbuf(cbuf->body, cbuf->bsize, 1);
    }
    return xmldoc;
}
```

5.3.2.2 struct curlbuf* curl_geturl (const char * *def_url,* struct basic_auth * *bauth,* curl_authcb *authcb,* void * *auth_data*) [read]

Definition at line 254 of file curl.c.

```
                                        {
    return curl_sendurl(def_url, bauth, NULL, authcb, auth_data);
}
```

5.3.2.3 struct basic_auth* curl_newauth (const char * *user,* const char * *passwd*) [read]

Definition at line 290 of file curl.c.

References objalloc(), basic_auth::passwd, and basic_auth::user.

```
                                                                        {
    struct basic_auth *bauth;

    if (!(bauth = (struct basic_auth *)objalloc(sizeof(*bauth
      ), curl_freeauth))) {
        return NULL;
    }
    if (user) {
        bauth->user = strdup(user);
    } else {
        bauth->user = strdup("");
    }
    if (passwd) {
        bauth->passwd = strdup(passwd);
    } else {
        bauth->passwd = strdup("");
    }
    return bauth;
}
```

5.3.2.4 struct curl_post* curl_newpost (void) [read]

Definition at line 316 of file curl.c.

References curl_post::first, free_post(), curl_post::last, and objalloc().

```
                                          {
    struct curl_post *post;
    if (!(post = objalloc(sizeof(*post), free_post))) {
        return NULL;
    }
    post->first = NULL;
    post->last = NULL;
    return post;
}
```

5.3.2.5 void curl_postitem (struct curl_post * *post,* const char * *name,* const char * *item*)

Definition at line 326 of file curl.c.

References curl_post::first, curl_post::last, objlock(), and objunlock().

```
      {
    if (!name || !item) {
        return;
    }
    objlock(post);
    curl_formadd(&post->first, &post->last,
        CURLFORM_COPYNAME, name,
        CURLFORM_COPYCONTENTS, item,
        CURLFORM_END);
    objunlock(post);
}
```

5.3.2.6 struct curlbuf* curl_posturl (const char * *def_url,* struct basic_auth * *bauth,* struct curl_post * *post,* curl_authcb *authcb,* void * *auth_data*) [read]

Definition at line 258 of file curl.c.

```
                                                                    {
    return curl_sendurl(def_url, bauth, post, authcb, auth_data);
}
```

5.3.2.7 void curl_setauth_cb (curl_authcb *auth_cb,* void * *data*)

Definition at line 397 of file curl.c.

References free_curlpassword(), objalloc(), objref(), and objunref().

```
                                                          {
    if (curlpassword) {
        objunref(curlpassword);
        curlpassword = NULL;
    }

    if (!(curlpassword = objalloc(sizeof(*curlpassword),
      free_curlpassword))) {
        return;
    }

    curlpassword->authcb = auth_cb;
    if (data && objref(data)) {
        curlpassword->data = data;
    }
}
```

5.3.2.8 void curl_setprogress (curl_progress_func *cb,* curl_progress_pause *p_cb,* curl_progress_newdata *d_cb,* void * *data*)

Definition at line 373 of file curl.c.

References free_progress(), objalloc(), objref(), and objunref().

```
                                        {
    if (curlprogress) {
        objunref(curlprogress);
        curlprogress = NULL;
    }

    if (!(curlprogress = objalloc(sizeof(*curlprogress), free_progress
      ))) {
        return;
    }
    curlprogress->cb = cb;
    curlprogress->d_cb = d_cb;
    curlprogress->p_cb = p_cb;
    if (data && objref(data)) {
        curlprogress->data = data;
    }
}
```

5.3.2.9 struct curlbuf* curl_ungzip (struct curlbuf * *cbuf*) [read]

Definition at line 262 of file curl.c.

References curlbuf::body, curlbuf::bsize, gzinflatebuf(), and is_gzip().

Referenced by curl_buf2xml().

```
                                                {
    uint8_t *gzbuf;
    uint32_t len;

    if (is_gzip((uint8_t *)cbuf->body, cbuf->bsize) &&
            ((gzbuf = gzinflatebuf((uint8_t *)cbuf->body, cbuf
      ->bsize, &len)))) {
        free(cbuf->body);
        cbuf->body = gzbuf;
        cbuf->bsize = len;
    }
    return cbuf;
}
```

5.3.2.10 void curlclose (void)

Definition at line 107 of file curl.c.

References objunref().

```
                    {
    objunref(curl_isinit);
    curl_isinit = NULL;
}
```

5.3.2.11 int curlinit (void)

Definition at line 79 of file curl.c.

References objalloc(), objlock(), objref(), objunlock(), and objunref().

Referenced by url_escape(), and url_unescape().

```
                  {
    if (curl_isinit) {
        return objref(curl_isinit);
    }

    if (!(curl_isinit = objalloc(sizeof(void *),curlfree))) {
        return 0;
    }

    objlock(curl_isinit);
    if (!(curl = curl_easy_init())) {
        objunlock(curl_isinit);
        objunref(curl_isinit);
        return 0;
    }

    curl_easy_setopt(curl, CURLOPT_SSL_VERIFYPEER, 0);
    curl_easy_setopt(curl, CURLOPT_NOSIGNAL, 1);
    curl_easy_setopt(curl, CURLOPT_COOKIEFILE, "");

    curl_easy_setopt(curl, CURLOPT_USERAGENT, "libcurl-agent/1.0 [Distro
       Solutions]");

    curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, bodytobuffer);
    curl_easy_setopt(curl, CURLOPT_HEADERFUNCTION, headertobuffer);
    objunlock(curl_isinit);
    return 1;
}
```

5.3.2.12 void free_curlpassword (void * *data*)

Definition at line 390 of file curl.c.

References objunref().

Referenced by curl_setauth_cb().

```
                                    {
    struct curl_password *cpwd = data;
    if (cpwd->data) {
        objunref(cpwd->data);
    }
}
```

5.3.2.13 void free_post (void * *data*)

Definition at line 309 of file curl.c.

References curl_post::first.

Referenced by curl_newpost().

```
                            {
    struct curl_post *post = data;
    if (post->first) {
        curl_formfree(post->first);
    }
}
```

5.3.2.14 void free_progress (void * *data*)

Definition at line 366 of file curl.c.

References objunref().

Referenced by curl_setprogress().

```
                                {
    struct curl_progress *prg = data;
    if (prg->data) {
        objunref(prg->data);
    }
}
```

5.3.2.15 char* url_escape (char * *url*)

Definition at line 338 of file curl.c.

References curlinit(), objlock(), objunlock(), and objunref().

```
                                    {
    char *esc;

    if (!curlinit()) {
        return NULL;
    }

    objlock(curl_isinit);
    esc = curl_easy_escape(curl, url, 0);
    objunlock(curl_isinit);
    objunref(curl_isinit);
    return esc;
}
```

5.3.2.16 char* url_unescape (char * *url*)

Definition at line 352 of file curl.c.

References curlinit(), objlock(), objunlock(), and objunref().

```
                                        {
    char *uesc;

    if (!curlinit()) {
        return NULL;
    }

    objlock(curl_isinit);
    uesc = curl_easy_unescape(curl, url, 0, 0);
    objunlock(curl_isinit);
    objunref(curl_isinit);
    return uesc;
}
```

5.4 File utility functions

Convinece wrappers arround stat.

Functions

- int is_file (const char *path)
- int is_dir (const char *path)
- int is_exec (const char *path)
- int mk_dir (const char *dir, mode_t mode, uid_t user, gid_t group)

5.4.1 Detailed Description

Convinece wrappers arround stat.

5.4.2 Function Documentation

5.4.2.1 int is_dir (const char * *path*)

Definition at line 51 of file fileutil.c.

```
                                    {
    struct stat sr;
    if (!stat(path, &sr) && S_ISDIR(sr.st_mode)) {
        return 1;
    } else {
        return 0;
    }
}
```

5.4.2.2 int is_exec (const char * *path*)

Definition at line 60 of file fileutil.c.

```
                                     {
    struct stat sr;
    if (!stat(path, &sr) && (S_IXUSR & sr.st_mode)) {
        return 1;
    } else {
        return 0;
    }
}
```

5.4.2.3 int is_file (const char * *path*)

Definition at line 42 of file fileutil.c.

```
                                     {
    struct stat sr;
    if (!stat(path, &sr)) {
        return 1;
    } else {
        return 0;
    }
}
```

5.4.2.4 int mk_dir (const char * *dir*, mode_t *mode*, uid_t *user*, gid_t *group*)

Definition at line 72 of file fileutil.c.

```
                                                                                {
#endif
    struct stat sr;

#ifdef __WIN32
    if ((stat(dir, &sr) && (errno == ENOENT)) && !mkdir(dir)) {
#else
    if ((stat(dir, &sr) && (errno == ENOENT)) && !mkdir(dir, mode) && !chown(
      dir, user, group)) {
#endif
        return 0;
    }
    return -1;
}
```

5.5 Burtle Bob hash algorythim.

Default init value for hash function.

Files

- file lookup3.c

  by Bob Jenkins, May 2006, Public Domain.

Macros

- #define JHASH_INITVAL 0xdeadbeef
- #define jenhash(key, length, initval) hashlittle(key, length, (initval) ? initval : JHASH_INITVAL);

  Define jenhash as hashlittle on big endian it should be hashbig.
- #define HASH_LITTLE_ENDIAN 0
- #define HASH_BIG_ENDIAN 0
- #define hashsize(n) ((uint32_t)1<<(n))
- #define hashmask(n) (hashsize(n)-1)
- #define rot(x, k) (((x)<<(k)) | ((x)>>(32-(k))))
- #define mix(a, b, c)

  mix 3 32-bit values reversibly
- #define final(a, b, c)

  final mixing of 3 32-bit values (a,b,c) into c

Functions

- uint32_t hashword (const uint32_t *k, size_t length, uint32_t initval)

  hash a variable-length key into a 32-bit value (Big Endian)
- void hashword2 (const uint32_t *k, size_t length, uint32_t *pc, uint32_t *pb)

  same as hashword(), but take two seeds and return two 32-bit values
- uint32_t hashlittle (const void *key, size_t length, uint32_t initval)

  hash a variable-length key into a 32-bit value (Little Endian)
- void hashlittle2 (const void *key, size_t length, uint32_t *pc, uint32_t *pb)

  return 2 32-bit hash values.
- uint32_t hashbig (const void *key, size_t length, uint32_t initval)

  This is the same as hashword() on big-endian machines.

5.5.1 Detailed Description

Default init value for hash function. lookup3.c, by Bob Jenkins, May 2006, Public Domain.

5.5.2 Macro Definition Documentation

5.5.2.1 #define final(*a, b, c*)

Value:

```
{ \
  c ^= b; c -= rot(b,14); \
  a ^= c; a -= rot(c,11); \
  b ^= a; b -= rot(a,25); \
  c ^= b; c -= rot(b,16); \
  a ^= c; a -= rot(c,4);  \
  b ^= a; b -= rot(a,14); \
  c ^= b; c -= rot(b,24); \
}
```

final mixing of 3 32-bit values (a,b,c) into c

```
--------------------------------------------------------------------------------
final -- final mixing of 3 32-bit values (a,b,c) into c

Pairs of (a,b,c) values differing in only a few bits will usually
produce values of c that look totally different.  This was tested for
* pairs that differed by one bit, by two bits, in any combination
  of top bits of (a,b,c), or in any combination of bottom bits of
  (a,b,c).
* "differ" is defined as +, -, ^, or ~^.  For + and -, I transformed
  the output delta to a Gray code (a^(a>>1)) so a string of 1's (as
  is commonly produced by subtraction) look like a single 1-bit
  difference.
* the base values were pseudorandom, all zero but one bit set, or
  all zero plus a counter that starts at zero.

These constants passed:
 14 11 25 16 4 14 24
 12 14 25 16 4 14 24
and these came close:
  4  8 15 26 3 22 24
 10  8 15 26 3 22 24
 11  8 15 26 3 22 24
--------------------------------------------------------------------------------
```

Definition at line 160 of file lookup3.c.

5.5.2.2 #define HASH_BIG_ENDIAN 0

Definition at line 73 of file lookup3.c.

Referenced by hashbig().

5.5.2.3 #define HASH_LITTLE_ENDIAN 0

Definition at line 72 of file lookup3.c.

Referenced by hashlittle(), and hashlittle2().

5.5.2.4 #define hashmask(*n*) (hashsize(n)-1)

Definition at line 77 of file lookup3.c.

5.5.2.5 #define hashsize(*n*) ((uint32_t)1<<(n))

Definition at line 76 of file lookup3.c.

5.5.2.6 #define jenhash(*key, length, initval*) hashlittle(key, length, (initval) ? initval : JHASH_INITVAL);

Define jenhash as hashlittle on big endian it should be hashbig.

Definition at line 639 of file dtsapp.h.

Referenced by attr_hash(), ldapattr_hash(), modify_hash(), node_hash(), searchresults_hash(), and xslt_hash().

5.5.2.7 #define JHASH_INITVAL 0xdeadbeef

Definition at line 637 of file dtsapp.h.

5.5.2.8 #define mix(*a, b, c*)

Value:

```
{ \
  a -= c;  a ^= rot(c, 4);  c += b; \
  b -= a;  b ^= rot(a, 6);  a += c; \
  c -= b;  c ^= rot(b, 8);  b += a; \
  a -= c;  a ^= rot(c,16);  c += b; \
  b -= a;  b ^= rot(a,19);  a += c; \
  c -= b;  c ^= rot(b, 4);  b += a; \
}
```

mix 3 32-bit values reversibly

```
--------------------------------------------------------------------------------
mix -- mix 3 32-bit values reversibly.

This is reversible, so any information in (a,b,c) before mix() is
still in (a,b,c) after mix().

If four pairs of (a,b,c) inputs are run through mix(), or through
mix() in reverse, there are at least 32 bits of the output that
are sometimes the same for one pair and different for another pair.
This was tested for:
* pairs that differed by one bit, by two bits, in any combination
  of top bits of (a,b,c), or in any combination of bottom bits of
  (a,b,c).
* "differ" is defined as +, -, ^, or ~^.  For + and -, I transformed
  the output delta to a Gray code (a^(a>>1)) so a string of 1's (as
  is commonly produced by subtraction) look like a single 1-bit
  difference.
* the base values were pseudorandom, all zero but one bit set, or
  all zero plus a counter that starts at zero.

Some k values for my "a-=c; a^=rot(c,k); c+=b;" arrangement that
satisfy this are
    4  6  8 16 19  4
    9 15  3 18 27 15
   14  9  3  7 17  3
Well, "9 15 3 18 27 15" didn't quite get 32 bits diffing
for "differ" defined as + with a one-bit base and a two-bit delta.  I
used http://burtleburtle.net/bob/hash/avalanche.html to choose
the operations, constants, and arrangements of the variables.

This does not achieve avalanche.  There are input bits of (a,b,c)
that fail to affect some output bits of (a,b,c), especially of a.  The
most thoroughly mixed value is c, but it doesn't really even achieve
avalanche in c.

This allows some parallelism.  Read-after-writes are good at doubling
the number of bits affected, so the goal of mixing pulls in the opposite
direction as the goal of parallelism.  I did what I could.  Rotates
seem to cost as much as shifts on every machine I could lay my hands
on, and rotates are much kinder to the top and bottom bits, so I used
rotates.
--------------------------------------------------------------------------------
```

Definition at line 125 of file lookup3.c.

Referenced by hashbig(), hashlittle(), hashlittle2(), hashword(), and hashword2().

5.5.2.9 #define rot(x, k) (((x)<<(k)) | ((x)>>(32-(k))))

Definition at line 78 of file lookup3.c.

5.5.3 Function Documentation

5.5.3.1 uint32_t hashbig (const void * key, size_t length, uint32_t initval)

This is the same as hashword() on big-endian machines.

```
 * hashbig():
 * This is the same as hashword() on big-endian machines.  It is different
 * from hashlittle() on all machines.  hashbig() takes advantage of
 * big-endian byte ordering.
```

Definition at line 864 of file lookup3.c.

References HASH_BIG_ENDIAN, and mix.

```
                                                                    {
    uint32_t a,b,c;
    union {
        const void *ptr;
        size_t i;
    } u; /* to cast key to (size_t) happily */

    /* Set up the internal state */
    a = b = c = 0xdeadbeef + ((uint32_t)length) + initval;

    u.ptr = key;
    if (HASH_BIG_ENDIAN && ((u.i & 0x3) == 0)) {
        const uint32_t *k = (const uint32_t *)key;         /* read 32-bit
       chunks */
#ifdef VALGRIND
        const uint8_t  *k8;
#endif
        /*------ all but last block: aligned reads and affect 32 bits of
       (a,b,c) */
        while (length > 12) {
            a += k[0];
            b += k[1];
            c += k[2];
            mix(a,b,c);
            length -= 12;
            k += 3;
        }

        /*---------------------------- handle the last (probably partial)
       block */
        /*
         * "k[2]<<8" actually reads beyond the end of the string, but
         * then shifts out the part it's not allowed to read.  Because the
         * string is aligned, the illegal read is in the same word as the
         * rest of the string.  Every machine with memory protection I've seen
         * does it on word boundaries, so is OK with this.  But VALGRIND will
         * still catch it and complain.  The masking trick does make the hash
         * noticably faster for short strings (like English words).
         */
#ifndef VALGRIND

        switch(length) {
            case 12:
                c+=k[2];
                b+=k[1];
                a+=k[0];
                break;
            case 11:
                c+=k[2]&0xffffff00;
                b+=k[1];
                a+=k[0];
                break;
            case 10:
                c+=k[2]&0xffff0000;
                b+=k[1];
                a+=k[0];
                break;
            case 9 :
                c+=k[2]&0xff000000;
```

```
                b+=k[1];
                a+=k[0];
                break;
            case 8 :
                b+=k[1];
                a+=k[0];
                break;
            case 7 :
                b+=k[1]&0xffffff00;
                a+=k[0];
                break;
            case 6 :
                b+=k[1]&0xffff0000;
                a+=k[0];
                break;
            case 5 :
                b+=k[1]&0xff000000;
                a+=k[0];
                break;
            case 4 :
                a+=k[0];
                break;
            case 3 :
                a+=k[0]&0xffffff00;
                break;
            case 2 :
                a+=k[0]&0xffff0000;
                break;
            case 1 :
                a+=k[0]&0xff000000;
                break;
            case 0 :
                return (c);              /* zero length strings require no
       mixing */
         }

#else  /* make valgrind happy */

         k8 = (const uint8_t *)k;
         switch(length) {                 /* all the case statements fall
       through */
            case 12:
                c+=k[2];
                b+=k[1];
                a+=k[0];
                break;
            case 11:
                c+=((uint32_t)k8[10])<<8;  /* fall through */
            case 10:
                c+=((uint32_t)k8[9])<<16;  /* fall through */
            case 9 :
                c+=((uint32_t)k8[8])<<24;  /* fall through */
            case 8 :
                b+=k[1];
                a+=k[0];
                break;
            case 7 :
                b+=((uint32_t)k8[6])<<8;   /* fall through */
            case 6 :
                b+=((uint32_t)k8[5])<<16;  /* fall through */
            case 5 :
                b+=((uint32_t)k8[4])<<24;  /* fall through */
            case 4 :
                a+=k[0];
                break;
            case 3 :
                a+=((uint32_t)k8[2])<<8;   /* fall through */
            case 2 :
                a+=((uint32_t)k8[1])<<16;  /* fall through */
            case 1 :
                a+=((uint32_t)k8[0])<<24;
                break;
            case 0 :
                return c;
         }

#endif /* !VALGRIND */

    } else {                        /* need to read the key one byte at a time
       */
         const uint8_t *k = (const uint8_t *)key;

         /*--------------- all but the last block: affect some 32 bits of
       (a,b,c) */
         while (length > 12) {
            a += ((uint32_t)k[0])<<24;
            a += ((uint32_t)k[1])<<16;
```

```
            a += ((uint32_t)k[2])<<8;
            a += ((uint32_t)k[3]);
            b += ((uint32_t)k[4])<<24;
            b += ((uint32_t)k[5])<<16;
            b += ((uint32_t)k[6])<<8;
            b += ((uint32_t)k[7]);
            c += ((uint32_t)k[8])<<24;
            c += ((uint32_t)k[9])<<16;
            c += ((uint32_t)k[10])<<8;
            c += ((uint32_t)k[11]);
            mix(a,b,c);
            length -= 12;
            k += 12;
        }

        /*------------------------------- last block: affect all 32 bits of
       (c) */
        switch(length) {                   /* all the case statements fall
       through */
            case 12:
                c+=k[11];
                /* no break */
            case 11:
                c+=((uint32_t)k[10])<<8;
                /* no break */
            case 10:
                c+=((uint32_t)k[9])<<16;
                /* no break */
            case 9 :
                c+=((uint32_t)k[8])<<24;
                /* no break */
            case 8 :
                b+=k[7];
                /* no break */
            case 7 :
                b+=((uint32_t)k[6])<<8;
                /* no break */
            case 6 :
                b+=((uint32_t)k[5])<<16;
                /* no break */
            case 5 :
                b+=((uint32_t)k[4])<<24;
                /* no break */
            case 4 :
                a+=k[3];
                /* no break */
            case 3 :
                a+=((uint32_t)k[2])<<8;
                /* no break */
            case 2 :
                a+=((uint32_t)k[1])<<16;
                /* no break */
            case 1 :
                a+=((uint32_t)k[0])<<24;
                break;
            case 0 :
                return (c);
        }
    }

    final(a,b,c);
    return (c);
}
```

5.5.3.2 uint32_t hashlittle (const void * *key*, size_t *length*, uint32_t *initval*)

hash a variable-length key into a 32-bit value (Little Endian)

```
--------------------------------------------------------------------------------
hashlittle() -- hash a variable-length key into a 32-bit value
  k       : the key (the unaligned variable-length array of bytes)
  length  : the length of the key, counting by bytes
  initval : can be any 4-byte value
Returns a 32-bit value.  Every bit of the key affects every bit of
the return value.  Two keys differing by one or two bits will have
totally different hash values.

The best hash table sizes are powers of 2.  There is no need to do
mod a prime (mod is sooo slow!).  If you need less than 32 bits,
use a bitmask.  For example, if you need only 10 bits, do
```

```
  h = (h & hashmask(10));
In which case, the hash table should have hashsize(10) elements.

If you are hashing n strings (uint8_t **)k, do it like this:
  for (i=0, h=0; i<n; ++i) h = hashlittle( k[i], len[i], h);

By Bob Jenkins, 2006.  bob_jenkins@burtleburtle.net.  You may use this
code any way you wish, private, educational, or commercial.  It's free.

Use for hash table lookup, or anything where one collision in 2^^32 is
acceptable.  Do NOT use for cryptographic purposes.
--------------------------------------------------------------------------------
```

Definition at line 300 of file lookup3.c.

References HASH_LITTLE_ENDIAN, and mix.

```
                                                                        {
    uint32_t a,b,c;                                          /* internal state
      */
    union {
        const void *ptr;
        size_t i;
    } u;     /* needed for Mac Powerbook G4 */

    /* Set up the internal state */
    a = b = c = 0xdeadbeef + ((uint32_t)length) + initval;

    u.ptr = key;
    if (HASH_LITTLE_ENDIAN && ((u.i & 0x3) == 0)) {
        const uint32_t *k = (const uint32_t *)key;         /* read 32-bit
       chunks */
#ifdef VALGRIND
        const uint8_t  *k8;
#endif
        /*------ all but last block: aligned reads and affect 32 bits of
       (a,b,c) */
        while (length > 12) {
            a += k[0];
            b += k[1];
            c += k[2];
            mix(a,b,c);
            length -= 12;
            k += 3;
        }

        /*----------------------------- handle the last (probably partial)
       block */
        /*
         * "k[2]&0xffffff" actually reads beyond the end of the string, but
         * then masks off the part it's not allowed to read.  Because the
         * string is aligned, the masked-off tail is in the same word as the
         * rest of the string.  Every machine with memory protection I've seen
         * does it on word boundaries, so is OK with this.  But VALGRIND will
         * still catch it and complain.  The masking trick does make the hash
         * noticably faster for short strings (like English words).
         */
#ifndef VALGRIND

        switch(length) {
            case 12:
                c+=k[2];
                b+=k[1];
                a+=k[0];
                break;
            case 11:
                c+=k[2]&0xffffff;
                b+=k[1];
                a+=k[0];
                break;
            case 10:
                c+=k[2]&0xffff;
                b+=k[1];
                a+=k[0];
                break;
            case 9 :
                c+=k[2]&0xff;
                b+=k[1];
                a+=k[0];
                break;
            case 8 :
                b+=k[1];
                a+=k[0];
```

```
            break;
        case 7 :
            b+=k[1]&0xffffff;
            a+=k[0];
            break;
        case 6 :
            b+=k[1]&0xffff;
            a+=k[0];
            break;
        case 5 :
            b+=k[1]&0xff;
            a+=k[0];
            break;
        case 4 :
            a+=k[0];
            break;
        case 3 :
            a+=k[0]&0xffffff;
            break;
        case 2 :
            a+=k[0]&0xffff;
            break;
        case 1 :
            a+=k[0]&0xff;
            break;
        case 0 :
            return (c);              /* zero length strings require no
    mixing */
     }

#else /* make valgrind happy */

     k8 = (const uint8_t *)k;
     switch(length) {
        case 12:
            c+=k[2];
            b+=k[1];
            a+=k[0];
            break;
        case 11:
            c+=((uint32_t)k8[10])<<16;  /* fall through */
        case 10:
            c+=((uint32_t)k8[9])<<8;    /* fall through */
        case 9 :
            c+=k8[8];                   /* fall through */
        case 8 :
            b+=k[1];
            a+=k[0];
            break;
        case 7 :
            b+=((uint32_t)k8[6])<<16;   /* fall through */
        case 6 :
            b+=((uint32_t)k8[5])<<8;    /* fall through */
        case 5 :
            b+=k8[4];                   /* fall through */
        case 4 :
            a+=k[0];
            break;
        case 3 :
            a+=((uint32_t)k8[2])<<16;   /* fall through */
        case 2 :
            a+=((uint32_t)k8[1])<<8;    /* fall through */
        case 1 :
            a+=k8[0];
            break;
        case 0 :
            return c;
     }

#endif /* !valgrind */

  } else
     if (HASH_LITTLE_ENDIAN && ((u.i & 0x1) == 0)) {
        const uint16_t *k = (const uint16_t *)key;         /* read 16-bit
    chunks */
        const uint8_t  *k8;

        /*--------------- all but last block: aligned reads and different
    mixing */
        while (length > 12) {
            a += k[0] + (((uint32_t)k[1])<<16);
            b += k[2] + (((uint32_t)k[3])<<16);
            c += k[4] + (((uint32_t)k[5])<<16);
            mix(a,b,c);
            length -= 12;
            k += 6;
        }
```

```
     /*---------------------------- handle the last (probably partial)
block */
     k8 = (const uint8_t *)k;
     switch(length) {
         case 12:
             c+=k[4]+(((uint32_t)k[5])<<16);
             b+=k[2]+(((uint32_t)k[3])<<16);
             a+=k[0]+(((uint32_t)k[1])<<16);
             break;
         case 11:
             c+=((uint32_t)k8[10])<<16;     /* fall through */
             /* no break */
         case 10:
             c+=k[4];
             b+=k[2]+(((uint32_t)k[3])<<16);
             a+=k[0]+(((uint32_t)k[1])<<16);
             break;
         case 9 :
             c+=k8[8];                      /* fall through */
             /* no break */
         case 8 :
             b+=k[2]+(((uint32_t)k[3])<<16);
             a+=k[0]+(((uint32_t)k[1])<<16);
             break;
         case 7 :
             b+=((uint32_t)k8[6])<<16;      /* fall through */
             /* no break */
         case 6 :
             b+=k[2];
             a+=k[0]+(((uint32_t)k[1])<<16);
             break;
         case 5 :
             b+=k8[4];                      /* fall through */
             /* no break */
         case 4 :
             a+=k[0]+(((uint32_t)k[1])<<16);
             break;
         case 3 :
             a+=((uint32_t)k8[2])<<16;      /* fall through */
             /* no break */
         case 2 :
             a+=k[0];
             break;
         case 1 :
             a+=k8[0];
             break;
         case 0 :
             return (c);                     /* zero length requires no
mixing */
     }

 } else {                        /* need to read the key one byte at a
time */
     const uint8_t *k = (const uint8_t *)key;

     /*--------------- all but the last block: affect some 32 bits of
(a,b,c) */
     while (length > 12) {
         a += k[0];
         a += ((uint32_t)k[1])<<8;
         a += ((uint32_t)k[2])<<16;
         a += ((uint32_t)k[3])<<24;
         b += k[4];
         b += ((uint32_t)k[5])<<8;
         b += ((uint32_t)k[6])<<16;
         b += ((uint32_t)k[7])<<24;
         c += k[8];
         c += ((uint32_t)k[9])<<8;
         c += ((uint32_t)k[10])<<16;
         c += ((uint32_t)k[11])<<24;
         mix(a,b,c);
         length -= 12;
         k += 12;
     }

     /*-------------------------------- last block: affect all 32 bits
of (c) */
     switch(length) {                  /* all the case statements fall
through */
         case 12:
             c+=((uint32_t)k[11])<<24;
             /* no break */
         case 11:
             c+=((uint32_t)k[10])<<16;
             /* no break */
         case 10:
```

```
                    c+=((uint32_t)k[9])<<8;
                    /* no break */
                case 9 :
                    c+=k[8];
                    /* no break */
                case 8 :
                    b+=((uint32_t)k[7])<<24;
                    /* no break */
                case 7 :
                    b+=((uint32_t)k[6])<<16;
                    /* no break */
                case 6 :
                    b+=((uint32_t)k[5])<<8;
                    /* no break */
                case 5 :
                    b+=k[4];
                    /* no break */
                case 4 :
                    a+=((uint32_t)k[3])<<24;
                    /* no break */
                case 3 :
                    a+=((uint32_t)k[2])<<16;
                    /* no break */
                case 2 :
                    a+=((uint32_t)k[1])<<8;
                    /* no break */
                case 1 :
                    a+=k[0];
                    break;
                case 0 :
                    return (c);
            }
        }

    final(a,b,c);
    return (c);
}
```

5.5.3.3 void hashlittle2 (const void * *key*, size_t *length*, uint32_t * *pc*, uint32_t * *pb*)

return 2 32-bit hash values.

```
 * hashlittle2: return 2 32-bit hash values
 *
 * This is identical to hashlittle(), except it returns two 32-bit hash
 * values instead of just one.  This is good enough for hash table
 * lookup with 2^^64 buckets, or if you want a second hash if you're not
 * happy with the first, or if you want a probably-unique 64-bit ID for
 * the key.  *pc is better mixed than *pb, so use *pc first.  If you want
 * a 64-bit value do something like "*pc + (((uint64_t)*pb)<<32)".
```

Definition at line 576 of file lookup3.c.

References HASH_LITTLE_ENDIAN, and mix.

```
                {      /* IN: secondary initval, OUT: secondary hash */
    uint32_t a,b,c;                                          /* internal state
      */
    union {
        const void *ptr;
        size_t i;
    } u;     /* needed for Mac Powerbook G4 */

    /* Set up the internal state */
    a = b = c = 0xdeadbeef + ((uint32_t)length) + *pc;
    c += *pb;

    u.ptr = key;
    if (HASH_LITTLE_ENDIAN && ((u.i & 0x3) == 0)) {
        const uint32_t *k = (const uint32_t *)key;         /* read 32-bit
       chunks */
#ifdef VALGRIND
        const uint8_t  *k8;
#endif

        /*------ all but last block: aligned reads and affect 32 bits of
       (a,b,c) */
        while (length > 12) {
```

```
            a += k[0];
            b += k[1];
            c += k[2];
            mix(a,b,c);
            length -= 12;
            k += 3;
        }

        /*-------------------------------- handle the last (probably partial)
       block */
        /*
         * "k[2]&0xffffff" actually reads beyond the end of the string, but
         * then masks off the part it's not allowed to read.  Because the
         * string is aligned, the masked-off tail is in the same word as the
         * rest of the string.  Every machine with memory protection I've seen
         * does it on word boundaries, so is OK with this.  But VALGRIND will
         * still catch it and complain.  The masking trick does make the hash
         * noticably faster for short strings (like English words).
         */
#ifndef VALGRIND

        switch(length) {
            case 12:
                c+=k[2];
                b+=k[1];
                a+=k[0];
                break;
            case 11:
                c+=k[2]&0xffffff;
                b+=k[1];
                a+=k[0];
                break;
            case 10:
                c+=k[2]&0xffff;
                b+=k[1];
                a+=k[0];
                break;
            case 9 :
                c+=k[2]&0xff;
                b+=k[1];
                a+=k[0];
                break;
            case 8 :
                b+=k[1];
                a+=k[0];
                break;
            case 7 :
                b+=k[1]&0xffffff;
                a+=k[0];
                break;
            case 6 :
                b+=k[1]&0xffff;
                a+=k[0];
                break;
            case 5 :
                b+=k[1]&0xff;
                a+=k[0];
                break;
            case 4 :
                a+=k[0];
                break;
            case 3 :
                a+=k[0]&0xffffff;
                break;
            case 2 :
                a+=k[0]&0xffff;
                break;
            case 1 :
                a+=k[0]&0xff;
                break;
            case 0 :
                *pc=c;
                *pb=b;
                return;  /* zero length strings require no mixing */
        }

#else /* make valgrind happy */

        k8 = (const uint8_t *)k;
        switch(length) {
            case 12:
                c+=k[2];
                b+=k[1];
                a+=k[0];
                break;
            case 11:
                c+=((uint32_t)k8[10])<<16;  /* fall through */
```

```
            case 10:
                c+=((uint32_t)k8[9])<<8;    /* fall through */
            case 9 :
                c+=k8[8];                   /* fall through */
            case 8 :
                b+=k[1];
                a+=k[0];
                break;
            case 7 :
                b+=((uint32_t)k8[6])<<16;   /* fall through */
            case 6 :
                b+=((uint32_t)k8[5])<<8;    /* fall through */
            case 5 :
                b+=k8[4];                   /* fall through */
            case 4 :
                a+=k[0];
                break;
            case 3 :
                a+=((uint32_t)k8[2])<<16;   /* fall through */
            case 2 :
                a+=((uint32_t)k8[1])<<8;    /* fall through */
            case 1 :
                a+=k8[0];
                break;
            case 0 :
                *pc=c;
                *pb=b;
                return;  /* zero length strings require no mixing */
        }

#endif /* !valgrind */

    } else
        if (HASH_LITTLE_ENDIAN && ((u.i & 0x1) == 0)) {
            const uint16_t *k = (const uint16_t *)key;         /* read 16-bit
       chunks */
            const uint8_t  *k8;

            /*--------------- all but last block: aligned reads and different
       mixing */
            while (length > 12) {
                a += k[0] + (((uint32_t)k[1])<<16);
                b += k[2] + (((uint32_t)k[3])<<16);
                c += k[4] + (((uint32_t)k[5])<<16);
                mix(a,b,c);
                length -= 12;
                k += 6;
            }

            /*----------------------------- handle the last (probably partial)
       block */
            k8 = (const uint8_t *)k;
            switch(length) {
                case 12:
                    c+=k[4]+(((uint32_t)k[5])<<16);
                    b+=k[2]+(((uint32_t)k[3])<<16);
                    a+=k[0]+(((uint32_t)k[1])<<16);
                    break;
                case 11:
                    c+=((uint32_t)k8[10])<<16;     /* fall through */
                    /* no break */
                case 10:
                    c+=k[4];
                    b+=k[2]+(((uint32_t)k[3])<<16);
                    a+=k[0]+(((uint32_t)k[1])<<16);
                    break;
                case 9 :
                    c+=k8[8];                      /* fall through */
                    /* no break */
                case 8 :
                    b+=k[2]+(((uint32_t)k[3])<<16);
                    a+=k[0]+(((uint32_t)k[1])<<16);
                    break;
                case 7 :
                    b+=((uint32_t)k8[6])<<16;      /* fall through */
                    /* no break */
                case 6 :
                    b+=k[2];
                    a+=k[0]+(((uint32_t)k[1])<<16);
                    break;
                case 5 :
                    b+=k8[4];                      /* fall through */
                    /* no break */
                case 4 :
                    a+=k[0]+(((uint32_t)k[1])<<16);
                    break;
                case 3 :
```

```
                a+=((uint32_t)k8[2])<<16;      /* fall through */
                /* no break */
            case 2 :
                a+=k[0];
                break;
            case 1 :
                a+=k8[0];
                break;
            case 0 :
                *pc=c;
                *pb=b;
                return;  /* zero length strings require no mixing */
        }

     } else {                        /* need to read the key one byte at a
    time */
        const uint8_t *k = (const uint8_t *)key;

        /*--------------- all but the last block: affect some 32 bits of
    (a,b,c) */
        while (length > 12) {
            a += k[0];
            a += ((uint32_t)k[1])<<8;
            a += ((uint32_t)k[2])<<16;
            a += ((uint32_t)k[3])<<24;
            b += k[4];
            b += ((uint32_t)k[5])<<8;
            b += ((uint32_t)k[6])<<16;
            b += ((uint32_t)k[7])<<24;
            c += k[8];
            c += ((uint32_t)k[9])<<8;
            c += ((uint32_t)k[10])<<16;
            c += ((uint32_t)k[11])<<24;
            mix(a,b,c);
            length -= 12;
            k += 12;
        }

        /*-------------------------------- last block: affect all 32 bits
    of (c) */
        switch(length) {                   /* all the case statements fall
    through */
            case 12:
                c+=((uint32_t)k[11])<<24;
                /* no break */
            case 11:
                c+=((uint32_t)k[10])<<16;
                /* no break */
            case 10:
                c+=((uint32_t)k[9])<<8;
                /* no break */
            case 9 :
                c+=k[8];
                /* no break */
            case 8 :
                b+=((uint32_t)k[7])<<24;
                /* no break */
            case 7 :
                b+=((uint32_t)k[6])<<16;
                /* no break */
            case 6 :
                b+=((uint32_t)k[5])<<8;
                /* no break */
            case 5 :
                b+=k[4];
                /* no break */
            case 4 :
                a+=((uint32_t)k[3])<<24;
                /* no break */
            case 3 :
                a+=((uint32_t)k[2])<<16;
                /* no break */
            case 2 :
                a+=((uint32_t)k[1])<<8;
                /* no break */
            case 1 :
                a+=k[0];
                break;
            case 0 :
                *pc=c;
                *pb=b;
                return;  /* zero length strings require no mixing */
        }
     }

    final(a,b,c);
    *pc=c;
```

```
    *pb=b;
}
```

5.5.3.4 uint32_t hashword (const uint32_t * *k*, size_t *length*, uint32_t *initval*)

hash a variable-length key into a 32-bit value (Big Endian)

```
--------------------------------------------------------------------
 This works on all machines.  To be useful, it requires
 -- that the key be an array of uint32_t's, and
 -- that the length be the number of uint32_t's in the key

 The function hashword() is identical to hashlittle() on little-endian
 machines, and identical to hashbig() on big-endian machines,
 except that the length has to be measured in uint32_ts rather than in
 bytes.  hashlittle() is more complicated than hashword() only because
 hashlittle() has to dance around fitting the key bytes into registers.
--------------------------------------------------------------------
```

Definition at line 184 of file lookup3.c.

References mix.

```
                          {       /* the previous hash, or an arbitrary
   value */
    uint32_t a,b,c;

    /* Set up the internal state */
    a = b = c = 0xdeadbeef + (((uint32_t)length)<<2) + initval;

    /*------------------------------------------------ handle most of the key
      */
    while (length > 3) {
        a += k[0];
        b += k[1];
        c += k[2];
        mix(a,b,c);
        length -= 3;
        k += 3;
    }

    /*------------------------------------------- handle the last 3 uint32_t's
      */
    switch(length) {                   /* all the case statements fall through
      */
        case 3 :
            c+=k[2];
            /* no break */
        case 2 :
            b+=k[1];
            /* no break */
        case 1 :
            a+=k[0];
            final(a,b,c);
            /* no break */
        case 0:     /* case 0: nothing left to add */
            break;
    }
    /*------------------------------------------------------ report the result
      */
    return (c);
}
```

5.5.3.5 void hashword2 (const uint32_t * *k*, size_t *length*, uint32_t * *pc*, uint32_t * *pb*)

same as hashword(), but take two seeds and return two 32-bit values

```
--------------------------------------------------------------------
hashword2() -- same as hashword(), but take two seeds and return two
32-bit values.  pc and pb must both be nonnull, and *pc and *pb must
both be initialized with seeds.  If you pass in (*pb)==0, the output
(*pc) will be the same as the return value from hashword().
--------------------------------------------------------------------
```

Definition at line 231 of file lookup3.c.

References mix.

```
                        {             /* IN: more seed OUT: secondary hash
       value */
    uint32_t a,b,c;

    /* Set up the internal state */
    a = b = c = 0xdeadbeef + ((uint32_t)(length<<2)) + *pc;
    c += *pb;

    /*-------------------------------------------------- handle most of the key
      */
    while (length > 3) {
        a += k[0];
        b += k[1];
        c += k[2];
        mix(a,b,c);
        length -= 3;
        k += 3;
    }

    /*------------------------------------------- handle the last 3 uint32_t's
      */
    switch(length) {                    /* all the case statements fall through
      */
        case 3 :
            c+=k[2];
            /* no break */
        case 2 :
            b+=k[1];
            /* no break */
        case 1 :
            a+=k[0];
            final(a,b,c);
            /* no break */
        case 0:     /* case 0: nothing left to add */
            break;
    }
    /*------------------------------------------------------ report the result
      */
    *pc=c;
    *pb=b;
}
```

5.6 Linux network interface functions

Implement various interface routines from libnetlink.

Data Structures

- struct iplink_req
- struct ipaddr_req

Functions

- void closenetlink ()
- int get_iface_index (const char *ifname)
- int delete_kernvlan (char *ifname, int vid)
- int create_kernvlan (char *ifname, unsigned short vid)
- int delete_kernmac (char *ifname)
- int create_kernmac (char *ifname, char *macdev, unsigned char *mac)
- int set_interface_flags (int ifindex, int set, int clear)
- int set_interface_addr (int ifindex, const unsigned char *hwaddr)
- int set_interface_name (int ifindex, const char *name)
- int interface_bind (char *iface, int protocol, int flags)
- int eui48to64 (unsigned char *mac48, unsigned char *eui64)
- int get_ip6_addrprefix (const char *iface, unsigned char *prefix)
- void randhwaddr (unsigned char *addr)
- int create_tun (const char *ifname, const unsigned char *hwaddr, int flags)
- int ifdown (const char *ifname, int flags)
- int ifup (const char *ifname, int flags)
- int ifrename (const char *oldname, const char *newname)
- int ifhwaddr (const char *ifname, unsigned char *hwaddr)
- int set_interface_ipaddr (char *ifname, char *ipaddr)

5.6.1 Detailed Description

Implement various interface routines from libnetlink.

5.6.2 Function Documentation

5.6.2.1 void closenetlink (void)

Definition at line 86 of file interface.c.

References objunref().

```
                        {
    if (nlh) {
        objunref(nlh);
    }
}
```

5.6.2.2 int create_kernmac (char * *ifname,* char * *macdev,* unsigned char * *mac*)

Definition at line 224 of file interface.c.

References addattr32(), addattr_l(), get_iface_index(), iplink_req::n, NLMSG_TAIL, objalloc(), objlock(), objref(), objunlock(), objunref(), randhwaddr(), rtnl_talk(), and strlenzero().

```
                                                                        {
    struct iplink_req *req;
    struct rtattr *data, *linkinfo;
    unsigned char lmac[ETH_ALEN];
    char *type = "macvlan";
    int ifindex, ret;

    if (strlenzero(ifname) || (strlen(ifname) > IFNAMSIZ) ||
            strlenzero(macdev) || (strlen(macdev) > IFNAMSIZ) ||
            (!objref(nlh) && !(nlh = nlhandle(0)))) {
        return (-1);
    }

    /*set the index of base interface*/
    if (!(ifindex = get_iface_index(ifname))) {
        objunref(nlh);
        return (-1);
    }

    if (!mac) {
        randhwaddr(lmac);
    } else {
        strncpy((char *)lmac, (char *)mac, ETH_ALEN);
    }

    if (!(req = objalloc(sizeof(*req), NULL))) {
        objunref(nlh);
        return (-1);
    }

    req->n.nlmsg_len = NLMSG_LENGTH(sizeof(struct ifinfomsg));
    req->n.nlmsg_type = RTM_NEWLINK;
    req->n.nlmsg_flags = NLM_F_CREATE | NLM_F_EXCL | NLM_F_REQUEST;

    /*config base/dev/mac*/
    addattr_l(&req->n, sizeof(*req), IFLA_LINK, &ifindex, 4);
    addattr_l(&req->n, sizeof(*req), IFLA_IFNAME, macdev, strlen(
      macdev));
    addattr_l(&req->n, sizeof(*req), IFLA_ADDRESS, lmac, ETH_ALEN);

    /*type*/
    linkinfo  = NLMSG_TAIL(&req->n);
    addattr_l(&req->n, sizeof(*req), IFLA_LINKINFO, NULL, 0);
    addattr_l(&req->n, sizeof(*req), IFLA_INFO_KIND, type, strlen(
      type));

    /*mode*/
    data = NLMSG_TAIL(&req->n);
    addattr_l(&req->n, sizeof(*req), IFLA_INFO_DATA, NULL, 0);
    addattr32(&req->n, sizeof(*req), IFLA_MACVLAN_MODE,
      MACVLAN_MODE_PRIVATE);
    data->rta_len = (char *)NLMSG_TAIL(&req->n) - (char *)data;
    linkinfo->rta_len = (char *)NLMSG_TAIL(&req->n) - (char *)
      linkinfo;

    objlock(nlh);
    ret = rtnl_talk(nlh, &req->n, 0, 0, NULL);
    objunlock(nlh);

    objunref(nlh);
    objunref(req);

    return (ret);
}
```

5.6.2.3 int create_kernvlan (char * *ifname,* unsigned short *vid*)

Definition at line 161 of file interface.c.

References addattr_l(), get_iface_index(), iplink_req::n, NLMSG_TAIL, objalloc(), objlock(), objref(), objunlock(), objunref(), rtnl_talk(), and strlenzero().

```
                                                                        {
```

```
    struct iplink_req *req;
    char iface[IFNAMSIZ+1];
    struct rtattr *data, *linkinfo;
    char *type = "vlan";
    int ifindex, ret;

    if (strlenzero(ifname) || (strlen(ifname) > IFNAMSIZ) ||
            (!objref(nlh) && !(nlh = nlhandle(0)))) {
        return (-1);
    }

    /*set the index of base interface*/
    if (!(ifindex = get_iface_index(ifname))) {
        objunref(nlh);
        return (-1);
    }

    if (!(req = objalloc(sizeof(*req), NULL))) {
        objunref(nlh);
        return (-1);
    }

    snprintf(iface, IFNAMSIZ, "%s.%i", ifname, vid);
    req->n.nlmsg_len = NLMSG_LENGTH(sizeof(struct ifinfomsg));
    req->n.nlmsg_type = RTM_NEWLINK;
    req->n.nlmsg_flags = NLM_F_CREATE | NLM_F_EXCL | NLM_F_REQUEST;

    /*config base/dev/mac*/
    addattr_l(&req->n, sizeof(*req), IFLA_LINK, &ifindex, sizeof(
      ifindex));
    addattr_l(&req->n, sizeof(*req), IFLA_IFNAME, iface, strlen(iface
      ));

    /*type*/
    linkinfo  = NLMSG_TAIL(&req->n);
    addattr_l(&req->n, sizeof(*req), IFLA_LINKINFO, NULL, 0);
    addattr_l(&req->n, sizeof(*req), IFLA_INFO_KIND, type, strlen(
      type));

    /*vid*/
    data = NLMSG_TAIL(&req->n);
    addattr_l(&req->n, sizeof(*req), IFLA_INFO_DATA, NULL, 0);
    addattr_l(&req->n, sizeof(*req), IFLA_VLAN_ID, &vid, sizeof(vid))
      ;

    data->rta_len = (char *)NLMSG_TAIL(&req->n) - (char *)data;
    linkinfo->rta_len = (char *)NLMSG_TAIL(&req->n) - (char *)
      linkinfo;

    objlock(nlh);
    ret = rtnl_talk(nlh, &req->n, 0, 0, NULL);
    objunlock(nlh);

    objunref(nlh);
    objunref(req);

    return (ret);
}
```

5.6.2.4 int create_tun (const char * *ifname*, const unsigned char * *hwaddr*, int *flags*)

Definition at line 465 of file interface.c.

References get_iface_index(), set_interface_addr(), and set_interface_flags().

```
        {
    struct ifreq ifr;
    int fd, ifindex;
    char *tundev = "/dev/net/tun";

    /* open the tun/tap clone dev*/
    if ((fd = open(tundev, O_RDWR)) < 0) {
        return (-1);
    }

    /* configure the device*/
    memset(&ifr, 0, sizeof(ifr));
    ifr.ifr_flags = flags;
    strncpy(ifr.ifr_name, ifname, IFNAMSIZ);
    if (ioctl(fd, TUNSETIFF, (void *)&ifr) < 0 ) {
        perror("ioctl(TUNSETIFF) failed\n");
```

```
        close(fd);
        return (-1);
    }

    if (!(ifindex = get_iface_index(ifname))) {
        return (-1);
    }

    /* set the MAC address*/
    if (hwaddr) {
        set_interface_addr(ifindex, hwaddr);
    }

    /*set the network dev up*/
    set_interface_flags(ifindex, IFF_UP | IFF_RUNNING |
      IFF_MULTICAST | IFF_BROADCAST, 0);

    return (fd);
}
```

5.6.2.5 int delete_kernmac (char * *ifname*)

Definition at line 219 of file interface.c.

```
                                        {

    return (delete_interface(ifname));
}
```

5.6.2.6 int delete_kernvlan (char * *ifname*, int *vid*)

Definition at line 150 of file interface.c.

```
                                                    {
    char iface[IFNAMSIZ+1];

    /*check ifname grab a ref to nlh or open it*/
    snprintf(iface, IFNAMSIZ, "%s.%i", ifname, vid);
    return (delete_interface(iface));
}
```

5.6.2.7 int eui48to64 (unsigned char * *mac48*, unsigned char * *eui64*)

Definition at line 416 of file interface.c.

Referenced by get_ip6_addrprefix().

```
                                                                    {
    eui64[0] = (mac48[0] & 0xFE) ^ 0x02; /*clear multicast bit and flip local
       asignment*/
    eui64[1] = mac48[1];
    eui64[2] = mac48[2];
    eui64[3] = 0xFF;
    eui64[4] = 0xFE;
    eui64[5] = mac48[3];
    eui64[6] = mac48[4];
    eui64[7] = mac48[5];

    return (0);
}
```

5.6.2.8 int get_iface_index (const char * *ifname*)

Definition at line 92 of file interface.c.

References ll_init_map(), ll_name_to_index(), objlock(), objref(), objunlock(), and objunref().

Referenced by create_kernmac(), create_kernvlan(), create_tun(), ifdown(), ifhwaddr(), ifrename(), ifup(), interface_bind(), and set_interface_ipaddr().

```
                                                {
    int ifindex;

    if (!objref(nlh) && !(nlh = nlhandle(0))) {
        return (0);
    }

    objlock(nlh);
    ll_init_map(nlh, 1);
    objunlock(nlh);

    ifindex = ll_name_to_index(ifname);

    objunref(nlh);
    return (ifindex);
}
```

5.6.2.9 int get_ip6_addrprefix (const char * *iface*, unsigned char * *prefix*)

Definition at line 433 of file interface.c.

References eui48to64(), ifhwaddr(), sha1sum2(), and tvtontp64().

```
                                                                        {
    uint64_t ntpts;
    unsigned char eui64[8];
    unsigned char sha1[20];
    unsigned char mac48[ETH_ALEN];
    struct timeval tv;

    if (ifhwaddr(iface, mac48)) {
        return (-1);
    }

    gettimeofday(&tv, NULL);
    ntpts = tvtontp64(&tv);

    eui48to64(mac48, eui64);
    sha1sum2(sha1, (void *)&ntpts, sizeof(ntpts), (void *)eui64, sizeof
      (eui64));

    prefix[0] = 0xFD; /*0xFC | 0x01 FC00/7 with local bit set [8th bit]*/
    memcpy(prefix + 1, sha1+15, 5); /*LSD 40 bits of the SHA hash*/

    return (0);
}
```

5.6.2.10 int ifdown (const char * *ifname*, int *flags*)

Definition at line 500 of file interface.c.

References get_iface_index(), and set_interface_flags().

Referenced by ifrename().

```
                                                {
    int ifindex;

    /*down the device*/
    if (!(ifindex = get_iface_index(ifname))) {
        return (-1);
    }

    /*set the network dev up*/
    set_interface_flags(ifindex, 0, IFF_UP | IFF_RUNNING |
      flags);

    return (0);
}
```

5.6.2.11 int ifhwaddr (const char * *ifname*, unsigned char * *hwaddr*)

Definition at line 541 of file interface.c.

References get_iface_index(), ll_index_to_addr(), objref(), objunref(), and strlenzero().

Referenced by get_ip6_addrprefix().

```
                                                                    {
    int ifindex;

    if (!hwaddr || strlenzero(ifname) || (strlen(ifname) > IFNAMSIZ)
      ||
            (!objref(nlh) && !(nlh = nlhandle(0)))) {
        return (-1);
    }

    /*set the index of base interface*/
    if (!(ifindex = get_iface_index(ifname))) {
        objunref(nlh);
        return (-1);
    }

    ll_index_to_addr(ifindex, hwaddr, ETH_ALEN);
    objunref(nlh);
    return (0);
}
```

5.6.2.12 int ifrename (const char * *oldname*, const char * *newname*)

Definition at line 528 of file interface.c.

References get_iface_index(), ifdown(), and set_interface_name().

```
                                                                {
    int ifindex;

    ifdown(oldname, 0);

    if (!(ifindex = get_iface_index(oldname))) {
        return (-1);
    }
    set_interface_name(ifindex, newname);

    return (0);
}
```

5.6.2.13 int ifup (const char * *ifname*, int *flags*)

Definition at line 514 of file interface.c.

References get_iface_index(), and set_interface_flags().

```
                                                {
    int ifindex;

    /*down the device*/
    if (!(ifindex = get_iface_index(ifname))) {
        return (-1);
    }

    /*set the network dev up*/
    set_interface_flags(ifindex, IFF_UP | IFF_RUNNING |
      flags, 0);

    return (0);
}
```

5.6.2.14 int interface_bind (char * *iface*, int *protocol*, int *flags*)

Definition at line 381 of file interface.c.

References get_iface_index(), and set_interface_flags().

```
                                                                    {
    struct sockaddr_ll sll;
    int proto = htons(protocol);
    int fd, ifindex;

    /*set the network dev up*/
    if (!(ifindex = get_iface_index(iface))) {
        return (-1);
    }
    set_interface_flags(ifindex, IFF_UP | IFF_RUNNING, 0);

    /* open network raw socket */
    if ((fd = socket(PF_PACKET, SOCK_RAW,  proto)) < 0) {
        return (-1);
    }

    /*bind to the interface*/
    memset(&sll, 0, sizeof(sll));
    sll.sll_family = PF_PACKET;
    sll.sll_protocol = proto;
    sll.sll_ifindex = ifindex;
    if (bind(fd, (struct sockaddr *)&sll, sizeof(sll)) < 0) {
        perror("bind failed");
        close(fd);
        return (-1);
    }

    return (fd);
}
```

5.6.2.15 void randhwaddr (unsigned char * *addr*)

Definition at line 459 of file interface.c.

References genrand().

Referenced by create_kernmac().

```
                                            {
    genrand(addr, ETH_ALEN);
    addr [0] &= 0xfe;       /* clear multicast bit */
    addr [0] |= 0x02;       /* set local assignment bit (IEEE802) */
}
```

5.6.2.16 int set_interface_addr (int *ifindex*, const unsigned char * *hwaddr*)

Definition at line 321 of file interface.c.

References addattr_l(), iplink_req::i, iplink_req::n, objalloc(), objlock(), objref(), objunlock(), objunref(), and rtnl_talk().

Referenced by create_tun().

```
                                                                    {
    struct iplink_req *req;

    if ((!objref(nlh) && !(nlh = nlhandle(0)))) {
        return (-1);
    }

    if (!(req = objalloc(sizeof(*req), NULL))) {
        objunref(nlh);
        return (-1);
    }

    req->n.nlmsg_len = NLMSG_LENGTH(sizeof(struct ifinfomsg));
    req->n.nlmsg_type = RTM_NEWLINK;
    req->n.nlmsg_flags = NLM_F_REQUEST;
    req->i.ifi_index = ifindex;

    /*config base/dev/mac*/
    addattr_l(&req->n, sizeof(*req), IFLA_ADDRESS, hwaddr, ETH_ALEN);

    objlock(nlh);
    rtnl_talk(nlh, &req->n, 0, 0, NULL);
```

```
    objunlock(nlh);

    objunref(nlh);
    objunref(req);
    return (0);
}
```

5.6.2.17 int set_interface_flags (int *ifindex,* int *set,* int *clear*)

Definition at line 285 of file interface.c.

References iplink_req::i, ll_index_to_flags(), iplink_req::n, objalloc(), objlock(), objref(), objunlock(), objunref(), and rtnl_talk().

Referenced by create_tun(), ifdown(), ifup(), and interface_bind().

```
{
    struct iplink_req *req;
    int flags;

    if (!objref(nlh) && !(nlh = nlhandle(0))) {
        return (-1);
    }

    flags = ll_index_to_flags(ifindex);

    flags |= set;
    flags &= ~(clear);

    if (!(req = objalloc(sizeof(*req), NULL))) {
        objunref(nlh);
        return (-1);
    }

    req->n.nlmsg_len = NLMSG_LENGTH(sizeof(struct ifinfomsg));
    req->n.nlmsg_type = RTM_NEWLINK;
    req->n.nlmsg_flags = NLM_F_REQUEST;

    /*config base/dev/mac*/
    req->i.ifi_index = ifindex;
    req->i.ifi_flags = flags;
    req->i.ifi_change = set | clear;

    objlock(nlh);
    rtnl_talk(nlh, &req->n, 0, 0, NULL);
    objunlock(nlh);

    objunref(nlh);
    objunref(req);
    return (0);
}
```

5.6.2.18 int set_interface_ipaddr (char * *ifname,* char * *ipaddr*)

Definition at line 561 of file interface.c.

References addattr32(), addattr_l(), inet_prefix::bitlen, inet_prefix::bytelen, inet_prefix::data, inet_prefix::family, get_iface_index(), get_prefix(), ipaddr_req::i, ipaddr_req::n, objalloc(), objlock(), objref(), objunlock(), objunref(), and rtnl_talk().

```
{
    struct ipaddr_req *req;
    inet_prefix lcl;
    int ifindex, bcast;

    if ((!objref(nlh) && !(nlh = nlhandle(0)))) {
        return (-1);
    }

    if (!(req = objalloc(sizeof(*req), NULL))) {
        objunref(nlh);
        return (-1);
    }

    /*set the index of base interface*/
```

```
    if (!(ifindex = get_iface_index(ifname))) {
        objunref(nlh);
        return (-1);
    }

    req->n.nlmsg_len = NLMSG_LENGTH(sizeof(struct ifaddrmsg));
    req->n.nlmsg_type = RTM_NEWADDR;
    req->n.nlmsg_flags = NLM_F_REQUEST | NLM_F_EXCL | NLM_F_CREATE;

    req->i.ifa_scope = RT_SCOPE_HOST;
    req->i.ifa_index = ifindex;

    get_prefix(&lcl, ipaddr, AF_UNSPEC);
    req->i.ifa_family = lcl.family;
    req->i.ifa_prefixlen = lcl.bitlen;

    addattr_l(&req->n, sizeof(*req), IFA_LOCAL, &lcl.data, lcl.
      bytelen);
    addattr_l(&req->n, sizeof(*req), IFA_ADDRESS, &lcl.data, lcl.
      bytelen);
    if (lcl.family == AF_INET) {
        bcast = htonl((1 << (32 - lcl.bitlen)) - 1);
        addattr32(&req->n, sizeof(*req), IFA_BROADCAST, lcl.data[
      0] | bcast);
    }

    objlock(nlh);
    rtnl_talk(nlh, &req->n, 0, 0, NULL);
    objunlock(nlh);

    objunref(nlh);
    objunref(req);
    return (0);
}
```

5.6.2.19 int set_interface_name (int *ifindex*, const char * *name*)

Definition at line 350 of file interface.c.

References addattr_l(), iplink_req::i, iplink_req::n, objalloc(), objlock(), objref(), objunlock(), objunref(), and rtnl_talk().

Referenced by ifrename().

```
                                                                {
    struct iplink_req *req;

    if ((!objref(nlh) && !(nlh = nlhandle(0)))) {
        return (-1);
    }

    if (!(req = objalloc(sizeof(*req), NULL))) {
        objunref(nlh);
        return (-1);
    }

    req->n.nlmsg_len = NLMSG_LENGTH(sizeof(struct ifinfomsg));
    req->n.nlmsg_type = RTM_NEWLINK;
    req->n.nlmsg_flags = NLM_F_REQUEST;
    req->i.ifi_index = ifindex;

    addattr_l(&req->n, sizeof(*req), IFLA_IFNAME, name, strlen((
      char *)name));

    objlock(nlh);
    rtnl_talk(nlh, &req->n, 0, 0, NULL);
    objunlock(nlh);

    objunref(nlh);
    objunref(req);
    return (0);
}
```

5.7 IPv4 and IPv6 functions

Helper functions for various calculations.

Data Structures

- struct pseudohdr

Enumerations

- enum ipversion { IP_PROTO_V4 = 4, IP_PROTO_V6 = 6 }

Functions

- int checkipv6mask (const char *ipaddr, const char *network, uint8_t bits)
- void ipv4tcpchecksum (uint8_t *pkt)
- void ipv4udpchecksum (uint8_t *pkt)
- void icmpchecksum (uint8_t *pkt)
- void ipv4checksum (uint8_t *pkt)
- int packetchecksumv4 (uint8_t *pkt)
- int packetchecksumv6 (uint8_t *pkt)
- int packetchecksum (uint8_t *pkt)
- const char * cidrtosn (int bitlen, const char *buf, int size)
- const char * getnetaddr (const char *ipaddr, int cidr, const char *buf, int size)
- const char * getfirstaddr (const char *ipaddr, int cidr, const char *buf, int size)
- const char * getbcaddr (const char *ipaddr, int cidr, const char *buf, int size)
- const char * getlastaddr (const char *ipaddr, int cidr, const char *buf, int size)
- uint32_t cidrcnt (int bitlen)
- int reservedip (const char *ipaddr)
- char * ipv6to4prefix (const char *ipaddr)
- int check_ipv4 (const char *ip, int cidr, const char *test)

5.7.1 Detailed Description

Helper functions for various calculations.

5.7.2 Enumeration Type Documentation

5.7.2.1 enum ipversion

Enumerator:

IP_PROTO_V4

IP_PROTO_V6

Definition at line 69 of file iputil.c.

```
                {
    IP_PROTO_V4 = 4,
    IP_PROTO_V6 = 6
};
```

5.7.3 Function Documentation

5.7.3.1 int check_ipv4 (const char * *ip*, int *cidr*, const char * *test*)

Definition at line 326 of file iputil.c.

```
                                                                    {
    uint32_t ip1, ip2;

    inet_pton(AF_INET, ip, &ip1);
    inet_pton(AF_INET, test, &ip2);

    ip1 = ntohl(ip1) >> (32-cidr);
    ip2 = ntohl(ip2) >> (32-cidr);

    if (!(ip1 ^ ip2)) {
        return 1;
    } else {
        return 0;
    }
}
```

5.7.3.2 int checkipv6mask (const char * *ipaddr*, const char * *network*, uint8_t *bits*)

Definition at line 41 of file iputil.c.

```
    {
    uint8_t cnt, bytelen, bitlen;
    uint32_t mask, res = 0;
    uint32_t *nw = (uint32_t *)network;
    uint32_t *ip = (uint32_t *)ipaddr;

    /*calculate significant bytes and bits outside boundry*/
    if ((bitlen = bits % 32)) {
        bytelen = (bits - bitlen) / 32;
        bytelen++;
    } else {
        bytelen = bits / 32;
    }

    /*end loop on first mismatch do not check last block*/
    for(cnt = 0; (!res && (cnt < (bytelen - 1))); cnt++) {
        res += nw[cnt] ^ ip[cnt];
    }

    /*process last block if no error sofar*/
    if (!res) {
        mask = (bitlen) ? htonl(~((1 << (32 - bitlen)) - 1)) : -1;
        res += (nw[cnt] & mask) ^ (ip[cnt] & mask);
    }

    return (res);
}
```

5.7.3.3 uint32_t cidrcnt (int *bitlen*)

Definition at line 269 of file iputil.c.

```
                                {
    if (bitlen) {
        return pow(2, (32-bitlen));
    } else {
        return 0xFFFFFFFF;
    }
}
```

5.7.3.4 const char* cidrtosn (int *bitlen*, const char * *buf*, int *size*)

Definition at line 184 of file iputil.c.

```
                                                                    {
    uint32_t nm;

    if (!buf) {
        return NULL;
    }

    if (bitlen) {
        nm = ~((1 << (32-bitlen))-1);
    } else {
        nm = 0;
    }

    nm = htonl(nm);
    return inet_ntop(AF_INET, &nm, (char *)buf, size);
}
```

5.7.3.5 const char* getbcaddr (const char * *ipaddr*, int *cidr*, const char * *buf*, int *size*)

Definition at line 238 of file iputil.c.

```
          {
    uint32_t ip, mask;

    inet_pton(AF_INET, ipaddr, &ip);
    if (cidr) {
        mask = (1 << (32-cidr))-1;
        ip = ntohl(ip);
        ip = (ip & ~mask) | mask;
        ip = htonl(ip);
    } else {
        ip = 0;
    }
    return inet_ntop(AF_INET, &ip, (char *)buf, size);
}
```

5.7.3.6 const char* getfirstaddr (const char * *ipaddr*, int *cidr*, const char * *buf*, int *size*)

Definition at line 219 of file iputil.c.

```
             {
    uint32_t ip;

    if (!buf) {
        return NULL;
    }

    inet_pton(AF_INET, ipaddr, &ip);
    if (cidr) {
        ip = ntohl(ip);
        ip = ip & ~((1 << (32-cidr))-1);
        ip++;
        ip = htonl(ip);
    } else {
        ip = 1;
    }
    return inet_ntop(AF_INET, &ip, (char *)buf, size);
}
```

5.7.3.7 const char* getlastaddr (const char * *ipaddr*, int *cidr*, const char * *buf*, int *size*)

Definition at line 253 of file iputil.c.

```
           {
    uint32_t ip, mask;

    inet_pton(AF_INET, ipaddr, &ip);
    if (cidr) {
        mask = (1 << (32-cidr))-1;
```

```
        ip = ntohl(ip);
        ip = (ip & ~mask) | mask;
        ip--;
        ip = htonl(ip);
    } else {
        ip = 0;
    }
    return inet_ntop(AF_INET, &ip, (char *)buf, size);
}
```

5.7.3.8 const char* getnetaddr (const char * *ipaddr,* int *cidr,* const char * *buf,* int *size*)

Definition at line 201 of file iputil.c.

```
          {
    uint32_t ip;

    if (!buf) {
        return NULL;
    }

    inet_pton(AF_INET, ipaddr, &ip);
    if (cidr) {
        ip = ntohl(ip);
        ip = ip & ~((1 << (32-cidr))-1);
        ip = htonl(ip);
    } else {
        ip = 0;
    }
    return inet_ntop(AF_INET, &ip, (char *)buf, size);
}
```

5.7.3.9 void icmpchecksum (uint8_t * *pkt*)

Definition at line 118 of file iputil.c.

References checksum().

Referenced by packetchecksumv4().

```
                                    {
    struct iphdr *ip = (struct iphdr *)pkt;
    struct icmphdr *icmp = (struct icmphdr *)(pkt + (4 * ip->ihl));

    icmp->checksum = 0;
    icmp->checksum = checksum(icmp, ntohs(ip->tot_len) - (ip->ihl *4));
}
```

5.7.3.10 void ipv4checksum (uint8_t * *pkt*)

Definition at line 126 of file iputil.c.

References checksum().

Referenced by packetchecksumv4().

```
                                    {
    struct iphdr *ip = (struct iphdr *)pkt;

    ip->check = 0;
    ip->check = checksum(ip, (4 * ip->ihl));
}
```

5.7.3.11 void ipv4tcpchecksum (uint8_t * *pkt*)

Definition at line 82 of file iputil.c.

References checksum(), checksum_add(), pseudohdr::daddr, pseudohdr::len, pseudohdr::proto, pseudohdr::saddr, and pseudohdr::zero.

Referenced by packetchecksumv4().

```
                                        {
    struct iphdr *ip = (struct iphdr *)pkt;
    struct tcphdr *tcp = (struct tcphdr *)(pkt + (4 * ip->ihl));
    uint16_t plen, csum;
    struct pseudohdr phdr;

    /* get tcp packet len*/
    plen = ntohs(ip->tot_len) - (4 * ip->ihl);
    tcp->check = 0;
    phdr.saddr = ip->saddr;
    phdr.daddr = ip->daddr;
    phdr.zero = 0;
    phdr.proto = ip->protocol;
    phdr.len = htons(plen);
    csum = checksum(&phdr, sizeof(phdr));
    tcp->check = checksum_add(csum, tcp, plen);
}
```

5.7.3.12 void ipv4udpchecksum (uint8_t * *pkt*)

Definition at line 100 of file iputil.c.

References checksum(), checksum_add(), pseudohdr::daddr, pseudohdr::len, pseudohdr::proto, pseudohdr::saddr, and pseudohdr::zero.

Referenced by packetchecksumv4().

```
                                        {
    struct iphdr *ip = (struct iphdr *)pkt;
    struct udphdr *udp = (struct udphdr *)(pkt + (4 * ip->ihl));
    uint16_t csum, plen;
    struct pseudohdr phdr;

    /* get tcp packet len*/
    plen = ntohs(ip->tot_len) - (4 * ip->ihl);
    udp->check = 0;
    phdr.saddr = ip->saddr;
    phdr.daddr = ip->daddr;
    phdr.zero = 0;
    phdr.proto = ip->protocol;
    phdr.len = htons(plen);
    csum = checksum(&phdr, sizeof(phdr));
    udp->check = checksum_add(csum, udp, plen);
}
```

5.7.3.13 char* ipv6to4prefix (const char * *ipaddr*)

Definition at line 311 of file iputil.c.

References malloc.

```
                                                {
    uint32_t ip;
    uint8_t *ipa;
    char *pre6;

    if (!inet_pton(AF_INET, ipaddr, &ip)) {
        return NULL;
    }

    pre6 = malloc(10);
    ipa=(uint8_t*)&ip;
    snprintf(pre6, 10, "%02x%02x:%02x%02x", ipa[0], ipa[1], ipa[2], ipa[3]);
    return pre6;
}
```

5.7.3.14 int packetchecksum (uint8_t * *pkt*)

Definition at line 171 of file iputil.c.

References IP_PROTO_V4, IP_PROTO_V6, and packetchecksumv4().

```
                                        {
    struct iphdr *ip = (struct iphdr *)pkt;

    switch(ip->version) {
        case IP_PROTO_V4:
            return (packetchecksumv4(pkt));
            break;
        case IP_PROTO_V6:
            break;
    }
    return (-1);
}
```

5.7.3.15 int packetchecksumv4 (uint8_t * *pkt*)

Definition at line 133 of file iputil.c.

References icmpchecksum(), ipv4checksum(), ipv4tcpchecksum(), and ipv4udpchecksum().

Referenced by packetchecksum().

```
                                          {
    struct iphdr *ip = (struct iphdr *)pkt;

    ipv4checksum(pkt);

    switch(ip->protocol) {
        case IPPROTO_ICMP:
            icmpchecksum(pkt);
            break;
        case IPPROTO_TCP:
            ipv4tcpchecksum(pkt);
            break;
        case IPPROTO_UDP:
            ipv4udpchecksum(pkt);
            break;
        default
                :
            return (-1);
    }
    return (0);
}
```

5.7.3.16 int packetchecksumv6 (uint8_t * *pkt*)

Definition at line 155 of file iputil.c.

```
                                          {
    struct iphdr *ip = (struct iphdr *)pkt;
    switch(ip->protocol) {
        case IPPROTO_ICMP:
            break;
        case IPPROTO_TCP:
            break;
        case IPPROTO_UDP:
            break;
        default
                :
            return (-1);
    }
    return (0);
}
```

5.7.3.17 int reservedip (const char * *ipaddr*)

Definition at line 277 of file iputil.c.

```
                                        {
    uint32_t ip;

    inet_pton(AF_INET, ipaddr, &ip);
    ip = ntohl(ip);

    if (!((0xe0000000 ^ ip) >> 28)) { /* 224/4*/
        return 1;
    } else if (!((0x00000000 ^ ip) >> 24)) { /* 0/8 */
        return 1;
    } else if (!((0x0a000000 ^ ip) >> 24)) { /* 10/8 */
        return 1;
    } else if (!((0x7f000000 ^ ip) >> 24)) { /* 127/8 */
        return 1;
    } else if (!((0x64400000 ^ ip) >> 22)) { /* 100.64/10 */
        return 1;
    } else if (!((0xac100000 ^ ip) >> 20)) { /* 172.16/12 */
        return 1;
    } else if (!((0xc6120000 ^ ip) >> 17)) { /* 198.18/15 */
        return 1;
    } else if (!((0xc0a80000 ^ ip) >> 16)) { /* 192.168/16 */
        return 1;
    } else if (!((0xa9fe0000 ^ ip) >> 16)) { /* 169.254/16 */
        return 1;
    } else if (!((0xc0000200 ^ ip) >> 8)) { /* 192.0.2/24 */
        return 1;
    } else if (!((0xc6336400 ^ ip) >> 8)) { /* 198.51.100/24 */
        return 1;
    } else if (!((0xcb007100 ^ ip) >> 8)) { /* 203.0.113/24 */
        return 1;
    }
    return 0;
}
```

5.8 XML Interface

Utilities for managing XML documents.

Data Structures

- struct xml_node_iter
- struct xml_search

Functions

- void free_buffer (void *data)
- int node_hash (const void *data, int key)
- int attr_hash (const void *data, int key)
- struct xml_doc * xml_loaddoc (const char *docfile, int validate)
- struct xml_doc * xml_loadbuf (const uint8_t *buffer, uint32_t len, int validate)
- struct xml_node * xml_nodetohash (struct xml_doc *xmldoc, xmlNodePtr node, const char *attrkey)
- struct xml_node * xml_gethash (struct xml_search *xpsearch, int i, const char *attrkey)
- struct xml_node * xml_getrootnode (struct xml_doc *xmldoc)
- struct xml_node * xml_getfirstnode (struct xml_search *xpsearch, void **iter)
- struct xml_node * xml_getnextnode (void *iter)
- struct bucket_list * xml_getnodes (struct xml_search *xpsearch)
- struct bucket_list * xml_setnodes (struct xml_search *xpsearch, const char *attrkey)
- struct xml_search * xml_xpath (struct xml_doc *xmldata, const char *xpath, const char *attrkey)
- int xml_nodecount (struct xml_search *xsearch)
- struct xml_node * xml_getnode (struct xml_search *xsearch, const char *key)
- const char * xml_getattr (struct xml_node *xnode, const char *attr)
- const char * xml_getrootname (struct xml_doc *xmldoc)
- void xml_modify (struct xml_doc *xmldoc, struct xml_node *xnode, const char *value)
- void xml_setattr (struct xml_doc *xmldoc, struct xml_node *xnode, const char *name, const char *value)
- void xml_createpath (struct xml_doc *xmldoc, const char *xpath)
- void xml_appendnode (struct xml_doc *xmldoc, const char *xpath, struct xml_node *child)
- struct xml_node * xml_addnode (struct xml_doc *xmldoc, const char *xpath, const char *name, const char *value, const char *attrkey, const char *keyval)
- void xml_unlink (struct xml_node *xnode)
- void xml_delete (struct xml_node *xnode)
- char * xml_getbuffer (void *buffer)
- void * xml_doctobuffer (struct xml_doc *xmldoc)
- void xml_init ()
- void xml_close ()
- void xml_savefile (struct xml_doc *xmldoc, const char *file, int format, int compress)
- void xml_modify2 (struct xml_search *xpsearch, struct xml_node *xnode, const char *value)

Variables

- int xmlLoadExtwDtdDefaultValue

5.8.1 Detailed Description

Utilities for managing XML documents.

5.8.2 Function Documentation

5.8.2.1 int attr_hash (const void * *data*, int *key*)

Definition at line 100 of file libxml2.c.

References jenhash, and xml_attr::name.

Referenced by xml_nodetohash().

```
                                          {
    int ret;
    const struct xml_attr *ai = data;
    const char *hashkey = (key) ? data : ai->name;

    ret = jenhash(hashkey, strlen(hashkey), 0);

    return(ret);
}
```

5.8.2.2 void free_buffer (void * *data*)

Definition at line 39 of file libxml2.c.

References xml_buffer::buffer.

Referenced by xml_doctobuffer(), and xslt_apply_buffer().

```
                                {
    struct xml_buffer *xb = data;
    xmlFree(xb->buffer);
}
```

5.8.2.3 int node_hash (const void * *data*, int *key*)

Definition at line 87 of file libxml2.c.

References jenhash, and xml_node::key.

Referenced by xml_setnodes().

```
                                          {
    int ret;
    const struct xml_node *ni = data;
    const char *hashkey = (key) ? data : ni->key;

    if (hashkey) {
        ret = jenhash(hashkey, strlen(hashkey), 0);
    } else {
        ret = jenhash(ni, sizeof(ni), 0);
    }
    return(ret);
}
```

5.8.2.4 struct xml_node* xml_addnode (struct xml_doc * *xmldoc*, const char * *xpath*, const char * *name*, const char * *value*, const char * *attrkey*, const char * *keyval*) [read]

Definition at line 571 of file libxml2.c.

References xml_doc::doc, objlock(), objref(), objunlock(), objunref(), and xml_nodetohash().

Referenced by xml_createpath().

```
                                                                    {
    struct xml_node *newnode;
    xmlNodePtr parent;
```

```
    xmlNodePtr child;
    xmlChar *encval;

    if (!objref(xmldoc)) {
        return NULL;
    }

    objlock(xmldoc);
    if (!(parent = xml_getparent(xmldoc, xpath))) {
        objunlock(xmldoc);
        objunref(xmldoc);
        return NULL;
    }

    encval = xmlEncodeSpecialChars(xmldoc->doc, (const xmlChar *)value)
      ;
    child = xmlNewDocNode(xmldoc->doc, NULL, (const xmlChar *)name,
      encval);
    xmlFree(encval);
    xmlAddChild(parent,child);

    if (attrkey && keyval) {
        encval = xmlEncodeSpecialChars(xmldoc->doc, (const xmlChar *)keyval)
      ;
        xmlSetProp(child, (const xmlChar *)attrkey, (const xmlChar *)encval);
        xmlFree(encval);
    }
    objunlock(xmldoc);

    if (!(newnode = xml_nodetohash(xmldoc, child, attrkey))) {
        objunref(xmldoc);
        return NULL;
    }

    objunref(xmldoc);

    return newnode;
}
```

5.8.2.5 void xml_appendnode (struct xml_doc * *xmldoc*, const char * *xpath*, struct xml_node * *child*)

Definition at line 553 of file libxml2.c.

References xml_node::nodeptr, objlock(), objref(), objunlock(), and objunref().

```
                    {
    xmlNodePtr parent;

    if (!objref(xmldoc)) {
        return;
    }

    objlock(xmldoc);
    if (!(parent = xml_getparent(xmldoc, xpath))) {
        objunlock(xmldoc);
        objunref(xmldoc);
    }

    xmlAddChild(parent,child->nodeptr);
    objunlock(xmldoc);
    objunref(xmldoc);
}
```

5.8.2.6 void xml_close ( )

Definition at line 660 of file libxml2.c.

References objunref().

```
                      {
    if (xml_has_init_parser) {
        objunref(xml_has_init_parser);
    }
}
```

5.8.2.7 void xml_createpath (struct xml_doc * *xmldoc,* const char * *xpath*)

Definition at line 440 of file libxml2.c.

References malloc, objlock(), objref(), objunlock(), objunref(), xml_doc::root, xml_addnode(), and xml_doc::xpathCtx.

```
                                                                        {
    struct xml_node *nn;
    xmlXPathObjectPtr xpathObj;
    char *lpath, *tok, *save, *cpath, *dup;
    const char *root = (char *)xmldoc->root->name;
    int len;


    if (!objref(xmldoc)) {
        return;
    }

    if (!(dup = strdup(xpath))) {
        objunref(xmldoc);
        return;
    }

    len = strlen(xpath)+1;
    if (!(cpath = malloc(len))) {
        free(dup);
        objunref(xmldoc);
        return;
    }
    if (!(lpath = malloc(len))) {
        free(dup);
        free(cpath);
        objunref(xmldoc);
        return;
    }

    cpath[0] = '\0';
    lpath[0] = '\0';

#ifndef __WIN32__
        for (tok = strtok_r(dup, "/", &save); tok ; tok = strtok_r(NULL, "/", &
      save)) {
#else
        for (tok = strtok_s(dup, "/", &save); tok ; tok = strtok_s(NULL, "/", &
      save)) {
#endif
        strcat(cpath,"/");
        strcat(cpath, tok);
        if (!strcmp(tok, root)) {
            strcat(lpath,"/");
            strcat(lpath, tok);
            continue;
        }

        objlock(xmldoc);
        if (!(xpathObj = xmlXPathEvalExpression((const xmlChar *)cpath, xmldoc
      ->xpathCtx))) {
            objunlock(xmldoc);
            free(lpath);
            free(cpath);
            free(dup);
            objunref(xmldoc);
            return;
        }
        objunlock(xmldoc);

        if (xmlXPathNodeSetIsEmpty(xpathObj->nodesetval)) {
            nn = xml_addnode(xmldoc, lpath, tok, NULL, NULL, NULL);
            objunref(nn);
        }

        xmlXPathFreeObject(xpathObj);
        strcat(lpath,"/");
        strcat(lpath, tok);
    }

    free(dup);
    free(lpath);
    free(cpath);
    objunref(xmldoc);
}
```

5.8.2.8 void xml_delete (struct xml_node * *xnode*)

Definition at line 617 of file libxml2.c.

References xml_node::nodeptr, objlock(), and objunlock().

```
                                                {
    objlock(xnode);
    xmlUnlinkNode(xnode->nodeptr);
    xmlFreeNode(xnode->nodeptr);
    xnode->nodeptr = NULL;
    objunlock(xnode);
}
```

5.8.2.9 void* xml_doctobuffer (struct xml_doc * *xmldoc*)

Definition at line 634 of file libxml2.c.

References xml_buffer::buffer, xml_doc::doc, free_buffer(), objalloc(), objlock(), objunlock(), and xml_buffer::size.

```
                                                    {
    struct xml_buffer *xmlbuf;

    if (!(xmlbuf = objalloc(sizeof(*xmlbuf),free_buffer))) {
        return NULL;
    }

    objlock(xmldoc);
    xmlDocDumpFormatMemory(xmldoc->doc, &xmlbuf->buffer, &xmlbuf->size
      , 1);
    objunlock(xmldoc);
    return xmlbuf;
}
```

5.8.2.10 const char* xml_getattr (struct xml_node * *xnode,* const char * *attr*)

Definition at line 389 of file libxml2.c.

References xml_node::attrs, bucket_list_find_key(), objunref(), and xml_attr::value.

```
                                                                        {
    struct xml_attr *ainfo;

    if (!xnode) {
        return NULL;
    }

    if ((ainfo = bucket_list_find_key(xnode->attrs,
      attr))) {
        objunref(ainfo);
        return ainfo->value;
    } else {
        return NULL;
    }
}
```

5.8.2.11 char* xml_getbuffer (void * *buffer*)

Definition at line 625 of file libxml2.c.

References xml_buffer::buffer.

```
                                            {
    struct xml_buffer *xb = buffer;

    if (!xb) {
        return NULL;
    }
    return (char *)xb->buffer;
}
```

5.8.2.12 struct xml_node* xml_getfirstnode (struct xml_search * *xpsearch,* void ** *iter*) [read]

Definition at line 267 of file libxml2.c.

References xml_node_iter::cnt, xml_node_iter::curpos, objalloc(), objlock(), objref(), objunlock(), objunref(), xml_gethash(), xml_nodecount(), and xml_node_iter::xsearch.

```
        {
    struct xml_node_iter *newiter;
    struct xml_node *xn;

    if (!objref(xpsearch)) {
        return NULL;
    }

    if (iter) {
        newiter = objalloc(sizeof(*newiter), free_iter);
        objlock(xpsearch);
        newiter->cnt = xml_nodecount(xpsearch);
        objunlock(xpsearch);
        newiter->curpos = 0;
        newiter->xsearch = xpsearch;
        objref(newiter->xsearch);
        *iter = newiter;
    }

    xn = xml_gethash(xpsearch, 0, NULL);
    objunref(xpsearch);
    return xn;
}
```

5.8.2.13 struct xml_node* xml_gethash (struct xml_search * *xpsearch,* int *i,* const char * *attrkey*) [read]

Definition at line 220 of file libxml2.c.

References objlock(), objref(), objunlock(), objunref(), xml_nodetohash(), xml_search::xmldoc, and xml_search::xpathObj.

Referenced by xml_getfirstnode(), xml_getnextnode(), and xml_setnodes().

```
          {
    xmlNodePtr node;
    xmlNodeSetPtr nodeset;
    struct xml_node *xn;

    if (!objref(xpsearch)) {
        return NULL;
    }

    objlock(xpsearch->xmldoc);
    objlock(xpsearch);
    if (!(nodeset = xpsearch->xpathObj->nodesetval)) {
        objunlock(xpsearch);
        objunlock(xpsearch->xmldoc);
        objunref(xpsearch);
        return NULL;
    }

    if (!(node = nodeset->nodeTab[i])) {
        objunlock(xpsearch);
        objunlock(xpsearch->xmldoc);
        objunref(xpsearch);
        return NULL;
    }
    xn = xml_nodetohash(xpsearch->xmldoc, node, attrkey);
    objunlock(xpsearch);
    objunlock(xpsearch->xmldoc);
    objunref(xpsearch);

    return xn;
}
```

5.8.2.14 struct xml_node* xml_getnextnode (void * *iter*) [read]

Definition at line 291 of file libxml2.c.

References xml_node_iter::cnt, xml_node_iter::curpos, objlock(), objref(), objunlock(), objunref(), xml_gethash(), and xml_node_iter::xsearch.

```
                                                    {
    struct xml_node_iter *xi = iter;
    struct xml_node *xn;

    if (!objref(xi->xsearch)) {
        return NULL;
    }

    objlock(xi);
    xi->curpos ++;
    if (xi->curpos >= xi->cnt) {
        objunlock(xi);
        objunref(xi->xsearch);
        return NULL;
    }
    xn = xml_gethash(xi->xsearch, xi->curpos, NULL);
    objunlock(xi);
    objunref(xi->xsearch);

    return xn;
}
```

5.8.2.15 struct xml_node* xml_getnode (struct xml_search * *xsearch,* const char * *key*) [read]

Definition at line 382 of file libxml2.c.

References bucket_list_find_key(), and xml_search::nodes.

```
        {
    if (!xsearch) {
        return NULL;
    }
    return bucket_list_find_key(xsearch->nodes, key
      );
}
```

5.8.2.16 struct bucket_list* xml_getnodes (struct xml_search * *xpsearch*) [read]

Definition at line 313 of file libxml2.c.

References xml_search::nodes.

```
                                                          {
    if (!xpsearch) {
        return NULL;
    }
    return xpsearch->nodes;
}
```

5.8.2.17 const char* xml_getrootname (struct xml_doc * *xmldoc*)

Definition at line 404 of file libxml2.c.

References xml_doc::root.

```
                                                    {
    if (xmldoc) {
        return (const char *)xmldoc->root->name;
    }
    return NULL;
}
```

5.8.2.18 struct xml_node* xml_getrootnode (struct xml_doc * *xmldoc*) [read]

Definition at line 258 of file libxml2.c.

References objlock(), objunlock(), xml_doc::root, and xml_nodetohash().

```
                                                                {
    struct xml_node *rn;

    objlock(xmldoc);
    rn = xml_nodetohash(xmldoc, xmldoc->root, NULL);
    objunlock(xmldoc);
    return rn;
}
```

5.8.2.19 void xml_init ( )

Definition at line 647 of file libxml2.c.

References free_parser(), objalloc(), and objref().

Referenced by xml_loadbuf(), and xml_loaddoc().

```
                    {
    if (!xml_has_init_parser) {
        xml_has_init_parser = objalloc(0, free_parser);
        xmlInitParser();
        LIBXML_TEST_VERSION
        xmlKeepBlanksDefault(0);
        xmlLoadExtDtdDefaultValue = 1;
        xmlSubstituteEntitiesDefault(1);
    } else {
        objref(xml_has_init_parser);
    }
}
```

5.8.2.20 struct xml_doc* xml_loadbuf (const uint8_t * *buffer,* uint32_t *len,* int *validate*) [read]

Definition at line 152 of file libxml2.c.

References xml_doc::doc, objalloc(), objunref(), and xml_init().

Referenced by curl_buf2xml().

```
          {
    struct xml_doc *xmldata;
    int flags;

    xml_init();

    if (!(xmldata = objalloc(sizeof(*xmldata), free_xmldata))) {
        return NULL;
    }

    if (validate) {
        flags = XML_PARSE_DTDLOAD | XML_PARSE_DTDVALID;
    } else {
        flags = XML_PARSE_DTDVALID;
    }

    if (!(xmldata->doc = xmlReadMemory((const char *)buffer, len, NULL, NULL
      , flags))) {
        objunref(xmldata);
        return NULL;
    }
    return xml_setup_parse(xmldata, 0);
}
```

5.8.2.21 struct xml_doc* xml_loaddoc (const char * *docfile,* int *validate*) [read]

Definition at line 135 of file libxml2.c.

References xml_doc::doc, objalloc(), objunref(), and xml_init().

```
                                                                    {
    struct xml_doc *xmldata;

    xml_init();

    if (!(xmldata = objalloc(sizeof(*xmldata), free_xmldata))) {
        return NULL;
    }

    if (!(xmldata->doc = xmlParseFile(docfile))) {
        objunref(xmldata);
        return NULL;
    }

    return xml_setup_parse(xmldata, validate);
}
```

5.8.2.22 void xml_modify (struct xml_doc * *xmldoc,* struct xml_node * *xnode,* const char * *value*)

Definition at line 411 of file libxml2.c.

References ALLOC_CONST, xml_doc::doc, xml_node::nodeptr, objlock(), objunlock(), and xml_node::value.

```
                {
    xmlChar *encval;
    xmlNodePtr node;

    objlock(xmldoc);
    node = xnode->nodeptr;
    encval = xmlEncodeSpecialChars(xmldoc->doc, (const xmlChar *)value);
    xmlNodeSetContent(node, encval);
    xmlFree(encval);
    encval = xmlNodeListGetString(xmldoc->doc, node->xmlChildrenNode, 1);
    objunlock(xmldoc);

    if (xnode->value) {
        free((void*)xnode->value);
    }
    ALLOC_CONST(xnode->value, (const char *)encval);
    xmlFree(encval);
}
```

5.8.2.23 void xml_modify2 (struct xml_search * *xpsearch,* struct xml_node * *xnode,* const char * *value*)

Definition at line 674 of file libxml2.c.

References xml_node::nodeptr, xml_buffer::size, and xml_search::xpathObj.

```
                    {
    xmlNodeSetPtr nodes;
    int size, i;

    if (!(nodes = xpsearch->xpathObj->nodesetval)) {
        return;
    }

    size = (nodes) ? nodes->nodeNr : 0;

    /*
     * http://www.xmlsoft.org/examples/xpath2.c
     * remove the reference to the modified nodes from the node set
     * as they are processed, if they are not namespace nodes.
    */
    for(i = size - 1; i >= 0; i--) {
        if (nodes->nodeTab[i] == xnode->nodeptr) {
            xmlNodeSetContent(nodes->nodeTab[i], (const xmlChar *)value);
```

```
            if (nodes->nodeTab[i]->type != XML_NAMESPACE_DECL) {
                nodes->nodeTab[i] = NULL;
            }
        }
    }
}
```

5.8.2.24 int xml_nodecount (struct xml_search * *xsearch*)

Definition at line 372 of file libxml2.c.

References xml_search::xpathObj.

Referenced by xml_getfirstnode(), and xml_setnodes().

```
                                                    {
    xmlNodeSetPtr nodeset;

    if (xsearch && xsearch->xpathObj && ((nodeset = xsearch->xpathObj
      ->nodesetval))) {
        return nodeset->nodeNr;
    } else {
        return 0;
    }
}
```

5.8.2.25 struct xml_node* xml_nodetohash (struct xml_doc * *xmldoc*, xmlNodePtr *node*, const char * *attrkey*) [read]

Definition at line 175 of file libxml2.c.

References addtobucket(), ALLOC_CONST, attr_hash(), xml_node::attrs, create_bucketlist(), xml_doc::doc, xml_node::key, xml_attr::name, xml_node::name, xml_node::nodeptr, objalloc(), objunref(), xml_attr::value, and xml_node::value.

Referenced by xml_addnode(), xml_gethash(), and xml_getrootnode().

```
                    {
    struct xml_node *ninfo;
    struct xml_attr *ainfo;
    xmlChar *xmlstr;
    xmlAttr *attrs;

    if (!(ninfo = objalloc(sizeof(*ninfo), free_xmlnode))) {
        return NULL;
    }
    ninfo->attrs = NULL;

    if (!(ninfo->attrs = create_bucketlist(0, attr_hash
      ))) {
        objunref(ninfo);
        return NULL;
    }

    ALLOC_CONST(ninfo->name, (const char *)node->name);
    xmlstr = xmlNodeListGetString(xmldoc->doc, node->xmlChildrenNode, 1);
    ALLOC_CONST(ninfo->value, (const char *)xmlstr);
    xmlFree(xmlstr);
    ninfo->nodeptr = node;

    attrs = node->properties;
    while(attrs && attrs->name && attrs->children) {
        if (!(ainfo = objalloc(sizeof(*ainfo), NULL))) {
            objunref(ninfo);
            return NULL;
        }
        ALLOC_CONST(ainfo->name, (const char *)attrs->name);
        xmlstr = xmlNodeListGetString(xmldoc->doc, attrs->children, 1);
        ALLOC_CONST(ainfo->value, (const char *)xmlstr);
        if (attrkey && !strcmp((const char *)attrs->name, (const char *)attrkey
      )) {
            ALLOC_CONST(ninfo->key, (const char *)xmlstr);
        }
```

```
        xmlFree(xmlstr);
        addtobucket(ninfo->attrs, ainfo);
        objunref(ainfo);
        attrs = attrs->next;
    }
    if (!attrkey && ninfo->value) {
        ALLOC_CONST(ninfo->key, ninfo->value);
    }
    return ninfo;
}
```

5.8.2.26 void xml_savefile (struct xml_doc * *xmldoc,* const char * *file,* int *format,* int *compress*)

Definition at line 666 of file libxml2.c.

References xml_doc::doc, objlock(), and objunlock().

```
                    {
    objlock(xmldoc);
    xmlSetDocCompressMode(xmldoc->doc, compress);
    xmlSaveFormatFile(file, xmldoc->doc, format);
    xmlSetDocCompressMode(xmldoc->doc, 0);
    objunlock(xmldoc);
}
```

5.8.2.27 void xml_setattr (struct xml_doc * *xmldoc,* struct xml_node * *xnode,* const char * *name,* const char * *value*)

Definition at line 430 of file libxml2.c.

References xml_doc::doc, xml_node::nodeptr, objlock(), and objunlock().

```
                                    {
    xmlChar *encval;

    objlock(xmldoc);
    encval = xmlEncodeSpecialChars(xmldoc->doc, (const xmlChar *)value)
      ;
    xmlSetProp(xnode->nodeptr, (const xmlChar *)name, (const xmlChar
       *)encval);
    objunlock(xmldoc);
    xmlFree(encval);
}
```

5.8.2.28 struct bucket_list* xml_setnodes (struct xml_search * *xpsearch,* const char * *attrkey*) [read]

Definition at line 320 of file libxml2.c.

References addtobucket(), create_bucketlist(), node_hash(), objunref(), xml_gethash(), and xml_nodecount().

Referenced by xml_xpath().

```
        {
    struct xml_node *ninfo;
    struct bucket_list *nodes;
    int cnt, i;

    if (!(nodes = create_bucketlist(2, node_hash))) {
        return NULL;
    }

    cnt = xml_nodecount(xpsearch);
    for(i=0; i < cnt; i++) {
        ninfo = xml_gethash(xpsearch, i, attrkey);
        if (!addtobucket(nodes, ninfo)) {
            objunref(ninfo);
            objunref(nodes);
            nodes = NULL;
            break;
```

```
        }
        objunref(ninfo);
    }
    return nodes;
}
```

5.8.2.29 void xml_unlink (struct xml_node * *xnode*)

Definition at line 611 of file libxml2.c.

References xml_node::nodeptr, objlock(), and objunlock().

```
                                                {
    objlock(xnode);
    xmlUnlinkNode(xnode->nodeptr);
    objunlock(xnode);
}
```

5.8.2.30 struct xml_search* xml_xpath (struct xml_doc * *xmldata,* const char * *xpath,* const char * *attrkey*) [read]

Definition at line 343 of file libxml2.c.

References xml_search::nodes, objalloc(), objlock(), objref(), objunlock(), objunref(), xml_setnodes(), xml_search::xmldoc, xml_doc::xpathCtx, and xml_search::xpathObj.

```
                            {
    struct xml_search *xpsearch;

    if (!objref(xmldata) || !(xpsearch = objalloc(sizeof(*
      xpsearch), free_xmlsearch))) {
        return NULL;
    }

    objlock(xmldata);
    xpsearch->xmldoc = xmldata;
    if (!(xpsearch->xpathObj = xmlXPathEvalExpression((const xmlChar *)
      xpath, xmldata->xpathCtx))) {
        objunlock(xmldata);
        objunref(xpsearch);
        return NULL;
    }

    if (xmlXPathNodeSetIsEmpty(xpsearch->xpathObj->nodesetval)) {
        objunlock(xmldata);
        objunref(xpsearch);
        return NULL;
    }
    objunlock(xmldata);

    if (!(xpsearch->nodes = xml_setnodes(xpsearch, attrkey)))
      {
        objunref(xpsearch);
        return NULL;
    }
    return xpsearch;
}
```

5.8.3 Variable Documentation

5.8.3.1 int xmlLoadExtwDtdDefaultValue

5.9 XSLT Interface

Utilities for managing XML documents.

Data Structures

- struct xslt_doc
- struct xslt_param

Functions

- void free_xsltdoc (void *data)
- void free_parser (void *data)
- int xslt_hash (const void *data, int key)
- struct xslt_doc * xslt_open (const char *xsltfile)
- void free_param (void *data)
- void xslt_addparam (struct xslt_doc *xsltdoc, const char *param, const char *value)
- void xslt_clearparam (struct xslt_doc *xsltdoc)
- void xslt_apply (struct xml_doc *xmldoc, struct xslt_doc *xsltdoc, const char *filename, int comp)
- void * xslt_apply_buffer (struct xml_doc *xmldoc, struct xslt_doc *xsltdoc)
- void xslt_init ()
- void xslt_close ()

5.9.1 Detailed Description

Utilities for managing XML documents.

5.9.2 Function Documentation

5.9.2.1 void free_param (void * *data*)

Definition at line 73 of file libxslt.c.

References xslt_param::name, and xslt_param::value.

Referenced by xslt_addparam().

```
                            {
    struct xslt_param *param = data;
    if (param->name) {
        free((void *)param->name);
    }
    if (param->value) {
        free((void *)param->value);
    }
}
```

5.9.2.2 void free_parser (void * *data*)

Definition at line 42 of file libxslt.c.

Referenced by xml_init(), and xslt_init().

```
                            {
    xsltCleanupGlobals();
    xmlCleanupParser();
}
```

5.9.2.3 void free_xsltdoc (void * *data*)

Definition at line 34 of file libxslt.c.

References xslt_doc::doc, objunref(), xslt_doc::params, and xslt_close().

Referenced by xslt_open().

```
                                {
    struct xslt_doc *xsltdoc = data;

    xsltFreeStylesheet(xsltdoc->doc);
    objunref(xsltdoc->params);
    xslt_close();
}
```

5.9.2.4 void xslt_addparam (struct xslt_doc * *xsltdoc,* const char * *param,* const char * *value*)

Definition at line 83 of file libxslt.c.

References addtobucket(), ALLOC_CONST, free_param(), malloc, xslt_param::name, objalloc(), objlock(), objref(), objunlock(), objunref(), xslt_doc::params, and xslt_param::value.

```
                {
    struct xslt_param *xparam;
    int size;

    if (!xsltdoc || !xsltdoc->params || !objref(xsltdoc) || !(
      xparam = objalloc(sizeof(*xparam), free_param))) {
        return;
    }

    size = strlen(value) + 3;
    ALLOC_CONST(xparam->name, param);
    xparam->value = malloc(size);
    snprintf((char *)xparam->value, size, "'%s'", value);
    objlock(xsltdoc);
    addtobucket(xsltdoc->params, xparam);
    objunlock(xsltdoc);
    objunref(xparam);
    objunref(xsltdoc);
}
```

5.9.2.5 void xslt_apply (struct xml_doc * *xmldoc,* struct xslt_doc * *xsltdoc,* const char * *filename,* int *comp*)

Definition at line 149 of file libxslt.c.

References xslt_doc::doc, xml_doc::doc, objlock(), objunlock(), objunref(), touch(), and xslt_clearparam().

```
                                  {
    const char **params = NULL;
    xmlDocPtr res;

    /* ref's xml/xslt locks xslt IF set*/
    if (!(params = xslt_params(xmldoc, xsltdoc))) {
        return;
    }

#ifndef __WIN32__
    touch(filename, 80, 80);
#else
    touch(filename);
#endif
    objlock(xmldoc);
    res = xsltApplyStylesheet(xsltdoc->doc, xmldoc->doc, params);
    xsltSaveResultToFilename(filename, res, xsltdoc->doc, comp);
    objunlock(xmldoc);
    objunref(xmldoc);
    objunlock(xsltdoc);

    free(params);
```

```
    xmlFreeDoc(res);
    xslt_clearparam(xsltdoc);
    objunref(xsltdoc);
}
```

5.9.2.6 void* xslt_apply_buffer (struct xml_doc * *xmldoc,* struct xslt_doc * *xsltdoc*)

Definition at line 176 of file libxslt.c.

References xml_buffer::buffer, xslt_doc::doc, xml_doc::doc, free_buffer(), objalloc(), objlock(), objunlock(), objunref(), xml_buffer::size, and xslt_clearparam().

```
        {
    struct xml_buffer *xmlbuf;
    const char **params;
    xmlDocPtr res;

    if (!(xmlbuf = objalloc(sizeof(*xmlbuf),free_buffer))) {
        return NULL;
    }

    if (!(params = xslt_params(xmldoc, xsltdoc))) {
        objunref(xmlbuf);
        return NULL;
    }

    objlock(xmldoc);
    res = xsltApplyStylesheet(xsltdoc->doc, xmldoc->doc, params);
    xsltSaveResultToString(&xmlbuf->buffer, &xmlbuf->size, res,
      xsltdoc->doc);
    objunlock(xmldoc);
    objunref(xmldoc);
    objunlock(xsltdoc);

    free(params);
    xmlFreeDoc(res);
    xslt_clearparam(xsltdoc);
    objunref(xsltdoc);

    return xmlbuf;
}
```

5.9.2.7 void xslt_clearparam (struct xslt_doc * *xsltdoc*)

Definition at line 102 of file libxslt.c.

References create_bucketlist(), objlock(), objunlock(), objunref(), xslt_doc::params, and xslt_hash().

Referenced by xslt_apply(), and xslt_apply_buffer().

```
                                                {
    if (!xsltdoc || !xsltdoc->params) {
        return;
    }

    objlock(xsltdoc);
    objunref(xsltdoc->params);
    xsltdoc->params = create_bucketlist(0, xslt_hash
      );
    objunlock(xsltdoc);
}
```

5.9.2.8 void xslt_close ( )

Definition at line 213 of file libxslt.c.

References objunref().

Referenced by free_xsltdoc().

```
                        {
    if (xslt_has_init_parser) {
        objunref(xslt_has_init_parser);
    }
}
```

5.9.2.9 int xslt_hash (const void * *data*, int *key*)

Definition at line 47 of file libxslt.c.

References jenhash, and xslt_param::name.

Referenced by xslt_clearparam(), and xslt_open().

```
                                        {
    int ret;
    const struct xslt_param *xp = data;
    const char *hashkey = (key) ? data : xp->name;

    if (hashkey) {
        ret = jenhash(hashkey, strlen(hashkey), 0);
    } else {
        ret = jenhash(xp, sizeof(xp), 0);
    }
    return(ret);
}
```

5.9.2.10 void xslt_init ( )

Definition at line 205 of file libxslt.c.

References free_parser(), objalloc(), and objref().

Referenced by xslt_open().

```
                        {
    if (!xslt_has_init_parser) {
        xslt_has_init_parser=objalloc(0, free_parser);
    } else {
        objref(xslt_has_init_parser);
    }
}
```

5.9.2.11 struct xslt_doc* xslt_open (const char * *xsltfile*) [read]

Definition at line 60 of file libxslt.c.

References create_bucketlist(), xslt_doc::doc, free_xsltdoc(), objalloc(), xslt_doc::params, xslt_hash(), and xslt_-init().

```
                                                    {
    struct xslt_doc *xsltdoc;

    if (!(xsltdoc = objalloc(sizeof(*xsltdoc), free_xsltdoc
      ))) {
        return NULL;
    }
    xslt_init();

    xsltdoc->doc = xsltParseStylesheetFile((const xmlChar *)xsltfile);
    xsltdoc->params = create_bucketlist(0, xslt_hash
      );
    return xsltdoc;
}
```

5.10 Referenced Objects

Utilities for managing referenced objects.

Data Structures

- struct ref_obj
- struct blist_obj
- struct bucket_list
- struct bucket_loop

Macros

- #define REFOBJ_MAGIC 0xdeadc0de
- #define refobj_offset sizeof(struct ref_obj);

Functions

- void * objalloc (int size, objdestroy destructor)
- int objref (void *data)
- int objunref (void *data)
- int objcnt (void *data)
- int objsize (void *data)
- int objlock (void *data)
- int objtrylock (void *data)
- int objunlock (void *data)
- void * create_bucketlist (int bitmask, blisthash hash_function)
- int addtobucket (struct bucket_list *blist, void *data)
- struct bucket_loop * init_bucket_loop (struct bucket_list *blist)
- void stop_bucket_loop (struct bucket_loop *bloop)
- void * next_bucket_loop (struct bucket_loop *bloop)
- void remove_bucket_item (struct bucket_list *blist, void *data)
- void remove_bucket_loop (struct bucket_loop *bloop)
- int bucket_list_cnt (struct bucket_list *blist)
- void * bucket_list_find_key (struct bucket_list *blist, const void *key)
- void bucketlist_callback (struct bucket_list *blist, blist_cb callback, void *data2)
- void * objchar (const char *orig)

5.10.1 Detailed Description

Utilities for managing referenced objects.

5.10.2 Macro Definition Documentation

5.10.2.1 #define REFOBJ_MAGIC 0xdeadc0de

Definition at line 35 of file refobj.c.

Referenced by objalloc(), objcnt(), objlock(), objref(), objsize(), objtrylock(), objunlock(), and objunref().

5.10.2.2 #define refobj_offset sizeof(struct ref_obj);

Definition at line 81 of file refobj.c.

Referenced by addtobucket(), objalloc(), objcnt(), objlock(), objref(), objsize(), objtrylock(), objunlock(), and objunref().

5.10.3 Function Documentation

5.10.3.1 int addtobucket (struct bucket_list * *blist,* void * *data*)

Definition at line 338 of file refobj.c.

References bucket_list::bucketbits, bucket_list::count, ref_obj::data, blist_obj::data, blist_obj::hash, bucket_list::list, bucket_list::locks, malloc, blist_obj::next, objlock(), objref(), objunlock(), objunref(), blist_obj::prev, refobj_offset, and bucket_list::version.

Referenced by add_radserver(), attr2bl(), dts_ldapsearch(), framework_mkthread(), new_modreq(), process_config(), radconnect(), rfc6296_map_add(), xml_nodetohash(), xml_setnodes(), and xslt_addparam().

```
                                                                {
    char *ptr = data;
    struct ref_obj *ref;
    struct blist_obj *lhead, *tmp;
    unsigned int hash, bucket;

    if (!objref(blist)) {
        return (0);
    }

    if (!objref(data)) {
        objunref(blist);
        return (0);
    }

    ptr = ptr - refobj_offset;
    ref = (struct ref_obj *)ptr;

    hash = gethash(blist, data, 0);
    bucket = ((hash >> (32 - blist->bucketbits)) & ((1 << blist->
      bucketbits) - 1));

    pthread_mutex_lock(&blist->locks[bucket]);
    lhead = blist->list[bucket];
    /*no head or non null head*/
    if (!lhead || lhead->prev) {
        if (!(tmp = malloc(sizeof(*tmp)))) {
            pthread_mutex_unlock(&blist->locks[bucket]);
            objunref(data);
            objunref(blist);
            return (0);
        }
        memset(tmp, 0, sizeof(*tmp));
        tmp->hash = hash;
        tmp->data = ref;

        /*there is no head*/
        if (!lhead) {
            blist->list[bucket] = tmp;
            tmp->prev = tmp;
            tmp->next = NULL;
            /*become new head*/
        } else
            if (hash < lhead->hash) {
                tmp->next = lhead;
                tmp->prev = lhead->prev;
                lhead->prev = tmp;
                blist->list[bucket] = tmp;
                /*new tail*/
            } else
                if (hash > lhead->prev->hash) {
                    tmp->prev = lhead->prev;
                    tmp->next = NULL;
                    lhead->prev->next = tmp;
                    lhead->prev = tmp;
                    /*insert entry*/
                } else {
                    lhead = blist_gotohash(lhead, hash, blist->bucketbits
```

```
      );
                    tmp->next = lhead->next;
                    tmp->prev = lhead;

                    if (lhead->next) {
                        lhead->next->prev = tmp;
                    } else {
                        blist->list[bucket]->prev = tmp;
                    }
                    lhead->next = tmp;
                }
    } else {
        /*set NULL head*/
        lhead->data = ref;
        lhead->prev = lhead;
        lhead->next = NULL;
        lhead->hash = hash;
    }

    blist->version[bucket]++;
    pthread_mutex_unlock(&blist->locks[bucket]);

    objlock(blist);
    blist->count++;
    objunlock(blist);
    objunref(blist);

    return (1);
}
```

5.10.3.2 int bucket_list_cnt (struct bucket_list * *blist*)

Definition at line 580 of file refobj.c.

References bucket_list::count, objlock(), and objunlock().

Referenced by add_radserver(), ldap_doadd(), and ldap_domodify().

```
                                                    {
    int ret = -1;

    if (blist) {
        objlock(blist);
        ret = blist->count;
        objunlock(blist);
    }
    return (ret);
}
```

5.10.3.3 void* bucket_list_find_key (struct bucket_list * *blist,* const void * *key*)

Definition at line 591 of file refobj.c.

References bucket_list::bucketbits, ref_obj::data, blist_obj::data, blist_obj::hash, bucket_list::list, bucket_list::locks, objref(), and objunref().

Referenced by get_config_category(), get_config_entry(), get_config_file(), getaddreq(), getmodreq(), ldap_-getattr(), ldap_getentry(), nfqueue_attach(), xml_getattr(), and xml_getnode().

```
                                                                    {
    struct blist_obj *entry;
    int hash, bucket;

    if (!blist) {
        return (NULL);
    }

    hash = gethash(blist, key, 1);
    bucket = ((hash >> (32 - blist->bucketbits)) & ((1 << blist->
      bucketbits) - 1));

    pthread_mutex_lock(&blist->locks[bucket]);
    entry = blist_gotohash(blist->list[bucket], hash + 1, blist->bucketbits
      );
    if (entry && entry->data) {
```

```
        objref(entry->data->data);
    } else
        if (!entry) {
            pthread_mutex_unlock(&blist->locks[bucket]);
            return NULL;
        }

    pthread_mutex_unlock(&blist->locks[bucket]);

    if (entry->data && (entry->hash == hash)) {
        return (entry->data->data);
    } else
        if (entry->data) {
            objunref(entry->data->data);
        }

    return NULL;
}
```

5.10.3.4 void bucketlist_callback (struct bucket_list * *blist,* blist_cb *callback,* void * *data2*)

Definition at line 624 of file refobj.c.

References init_bucket_loop(), next_bucket_loop(), objunref(), and stop_bucket_loop().

Referenced by config_cat_callback(), config_entry_callback(), config_file_callback(), and rfc6296_test().

```
                {
    struct bucket_loop *bloop;
    void *data;

    if (!blist || !callback) {
        return;
    }

    bloop = init_bucket_loop(blist);
    while(blist && bloop && (data = next_bucket_loop(bloop))) {
        callback(data, data2);
        objunref(data);
    }
    stop_bucket_loop(bloop);
}
```

5.10.3.5 void* create_bucketlist (int *bitmask,* blisthash *hash_function*)

Definition at line 268 of file refobj.c.

References bucket_list::bucketbits, malloc, and objalloc().

Referenced by add_radserver(), attr2bl(), dts_ldapsearch(), initconfigfiles(), ldap_addinit(), ldap_modifyinit(), radconnect(), rfc6296_map_add(), socketserver(), startthreads(), xml_nodetohash(), xml_setnodes(), xslt_clearparam(), and xslt_open().

```
                                                                {
    struct bucket_list *new;
    short int buckets, cnt;

    buckets = (1 << bitmask);

    /* allocate session bucket list memory*/
    if (!(new = objalloc(sizeof(*new) + (sizeof(void *) + sizeof(
      pthread_mutex_t) + sizeof(int)) * buckets, empty_buckets))) {
        return NULL;
    }

    /*initialise each bucket*/
    new->bucketbits = bitmask;
    new->list = (void *)((char *)new + sizeof(*new));
    for (cnt = 0; cnt < buckets; cnt++) {
        if ((new->list[cnt] = malloc(sizeof(*new->list[cnt])))) {
            memset(new->list[cnt], 0, sizeof(*new->list[cnt]));
        }
    }
```

```
    /*next pointer is pointer to locks*/
    new->locks = (void *)&new->list[buckets];
    for (cnt = 0; cnt < buckets; cnt++) {
        pthread_mutex_init(&new->locks[cnt], NULL);
    }

    /*Next up version array*/
    new->version = (void *)&new->locks[buckets];

    new->hash_func = hash_function;

    return (new);
}
```

5.10.3.6 struct bucket_loop* init_bucket_loop (struct bucket_list * *blist*) [read]

Definition at line 427 of file refobj.c.

References bucket_loop::blist, bucket_loop::bucket, blist_obj::hash, bucket_loop::head, bucket_loop::head_hash, bucket_list::list, bucket_list::locks, objalloc(), objref(), bucket_list::version, and bucket_loop::version.

Referenced by bucketlist_callback(), get_category_loop(), ldap_doadd(), and ldap_domodify().

```
                                                                    {
    struct bucket_loop *bloop = NULL;

    if (blist && (bloop = objalloc(sizeof(*bloop), NULL))) {
        objref(blist);
        bloop->blist = blist;
        bloop->bucket = 0;
        pthread_mutex_lock(&blist->locks[bloop->bucket]);
        bloop->head = blist->list[0];
        if (bloop->head) {
            bloop->head_hash = bloop->head->hash;
        };
        bloop->version = blist->version[0];
        pthread_mutex_unlock(&blist->locks[bloop->bucket]);
    }

    return (bloop);
}
```

5.10.3.7 void* next_bucket_loop (struct bucket_loop * *bloop*)

Definition at line 460 of file refobj.c.

References bucket_loop::blist, bucket_loop::bucket, bucket_list::bucketbits, bucket_loop::cur, bucket_loop::cur_hash, ref_obj::data, blist_obj::data, blist_obj::hash, bucket_loop::head, bucket_loop::head_hash, bucket_list::list, bucket_list::locks, blist_obj::next, objref(), blist_obj::prev, bucket_list::version, and bucket_loop::version.

Referenced by bucketlist_callback(), get_category_next(), ldap_doadd(), and ldap_domodify().

```
                                                          {
    struct bucket_list *blist = bloop->blist;
    struct ref_obj *entry = NULL;
    void *data = NULL;

    pthread_mutex_lock(&blist->locks[bloop->bucket]);
    if (bloop->head_hash && (blist->version[bloop->bucket
      ] != bloop->version)) {
        /* bucket has changed unexpectedly i need to ff/rew to hash*/
        bloop->head = blist_gotohash(blist->list[bloop->bucket],
      bloop->head_hash + 1, blist->bucketbits);
        /*if head has gone find next suitable ignore any added*/
        while (bloop->head && (bloop->head->hash < bloop->head_hash
      )) {
            bloop->head = bloop->head->next;
        }
    }

    while (!bloop->head || !bloop->head->prev) {
        pthread_mutex_unlock(&blist->locks[bloop->bucket]);
        bloop->bucket++;
        if (bloop->bucket < (1 << blist->bucketbits)) {
```

```
            pthread_mutex_lock(&blist->locks[bloop->bucket]);
            bloop->head = blist->list[bloop->bucket];
        } else {
            return NULL;
        }
    }

    if (bloop->head) {
        bloop->cur = bloop->head;
        entry = (bloop->head->data) ? bloop->head->data : NULL;
        data = (entry) ? entry->data : NULL;
        objref(data);
        bloop->head = bloop->head->next;
        bloop->head_hash = (bloop->head) ? bloop->head->hash
       : 0;
        bloop->cur_hash = (bloop->cur) ? bloop->cur->hash : 0
      ;
    }
    pthread_mutex_unlock(&blist->locks[bloop->bucket]);

    return (data);
}
```

5.10.3.8 void* objalloc (int *size,* objdestroy *destructor*)

Definition at line 83 of file refobj.c.

References ref_obj::cnt, ref_obj::data, ref_obj::destroy, ref_obj::lock, ref_obj::magic, malloc, REFOBJ_MAGIC, refobj_offset, and ref_obj::size.

Referenced by add_modifyval(), add_radserver(), attr2bl(), b64enc_buf(), create_bucketlist(), create_kernmac(), create_kernvlan(), curl_newauth(), curl_newpost(), curl_setauth_cb(), curl_setprogress(), curlinit(), dtls_listenssl(), dts_ldapsearch(), framework_mkcore(), framework_mkthread(), framework_unixsocket(), init_bucket_loop(), ldap_addinit(), ldap_connect(), ldap_encattr(), ldap_getent(), ldap_modifyinit(), ldap_saslbind(), ldap_simplebind(), make_socket(), new_modreq(), nfqueue_attach(), objchar(), radconnect(), rfc6296_map_add(), set_interface_addr(), set_interface_flags(), set_interface_ipaddr(), set_interface_name(), startthreads(), tlsaccept(), xml_doctobuffer(), xml_getfirstnode(), xml_init(), xml_loadbuf(), xml_loaddoc(), xml_nodetohash(), xml_xpath(), xslt_addparam(), xslt_apply_buffer(), xslt_init(), xslt_open(), and zcompress().

```
                                                    {
    struct ref_obj *ref;
    int asize;
    char *robj;

    asize  = size + refobj_offset;

    if ((robj = malloc(asize))) {
        memset(robj, 0, asize);
        ref = (struct ref_obj *)robj;
        if (!(ref->lock = malloc(sizeof(pthread_mutex_t)))) {
            free(robj);
            return NULL;
        }
        pthread_mutex_init(ref->lock, NULL);
        ref->magic = REFOBJ_MAGIC;
        ref->cnt = 1;
        ref->data = robj + refobj_offset;
        ref->size = size;
        ref->destroy = destructor;
        return (ref->data);
    }
    return NULL;
}
```

5.10.3.9 void* objchar (const char * *orig*)

Definition at line 640 of file refobj.c.

References objalloc().

```
                                        {
    int len = strlen(orig) + 1;
```

```
    void *nobj;

    if ((nobj = objalloc(len, NULL))) {
        memcpy(nobj, orig, len);
    }
    return nobj;
}
```

5.10.3.10 int objcnt (void * *data*)

Definition at line 171 of file refobj.c.

References ref_obj::cnt, ref_obj::data, ref_obj::lock, ref_obj::magic, REFOBJ_MAGIC, and refobj_offset.

Referenced by ldap_unref_attr(), and ldap_unref_entry().

```
                                {
    char *ptr = data;
    int ret = -1;
    struct ref_obj *ref;

    if (!data) {
        return (ret);
    }

    ptr = ptr - refobj_offset;
    ref = (struct ref_obj *)ptr;

    if (ref->magic == REFOBJ_MAGIC) {
        pthread_mutex_lock(ref->lock);
        ret = ref->cnt;
        pthread_mutex_unlock(ref->lock);
    }
    return (ret);
}
```

5.10.3.11 int objlock (void * *data*)

Definition at line 211 of file refobj.c.

References ref_obj::data, ref_obj::lock, ref_obj::magic, REFOBJ_MAGIC, and refobj_offset.

Referenced by addtobucket(), bucket_list_cnt(), create_kernmac(), create_kernvlan(), curl_postitem(), curlinit(), dtls_listenssl(), dtlshandltimeout(), dtlstimeout(), dts_ldapsearch(), dtsl_serveropts(), framework_mkthread(), get_iface_index(), ldap_attrvals(), ldap_count(), ldap_doadd(), ldap_domodify(), ldap_getattribute(), ldap_getdn(), ldap_getent(), ldap_saslbind(), ldap_simplebind(), nf_ctrack_delete(), nf_ctrack_dump(), nf_ctrack_nat(), nfqueue_attach(), remove_bucket_loop(), set_interface_addr(), set_interface_flags(), set_interface_ipaddr(), set_interface_name(), socketread_d(), socketserver(), socketwrite_d(), ssl_shutdown(), url_escape(), url_unescape(), xml_addnode(), xml_appendnode(), xml_createpath(), xml_delete(), xml_doctobuffer(), xml_getfirstnode(), xml_gethash(), xml_getnextnode(), xml_getrootnode(), xml_modify(), xml_savefile(), xml_setattr(), xml_unlink(), xml_xpath(), xslt_addparam(), xslt_apply(), xslt_apply_buffer(), and xslt_clearparam().

```
                                {
    char *ptr = data;
    struct ref_obj *ref;

    ptr = ptr - refobj_offset;
    ref = (struct ref_obj *)ptr;

    if (data && ref->magic == REFOBJ_MAGIC) {
        pthread_mutex_lock(ref->lock);
    }
    return (0);
}
```

5.10.3.12 int objref (void * *data*)

Definition at line 109 of file refobj.c.

References ref_obj::cnt, ref_obj::data, ref_obj::lock, ref_obj::magic, REFOBJ_MAGIC, and refobj_offset.

Referenced by addtobucket(), bucket_list_find_key(), create_kernmac(), create_kernvlan(), curl_setauth_cb(), curl_setprogress(), curlinit(), dts_ldapsearch(), framework_mkthread(), get_category_next(), get_config_category(), get_config_file(), get_iface_index(), ifhwaddr(), init_bucket_loop(), ldap_domodify(), ldap_rebind_proc(), ldap_saslbind(), ldap_simplebind(), ldap_simplerebind(), next_bucket_loop(), printgnu(), set_interface_addr(), set_interface_flags(), set_interface_ipaddr(), set_interface_name(), xml_addnode(), xml_appendnode(), xml_createpath(), xml_getfirstnode(), xml_gethash(), xml_getnextnode(), xml_init(), xml_xpath(), xslt_addparam(), and xslt_init().

```
                                {
    char *ptr = data;
    struct ref_obj *ref;
    int ret = 0;

    ptr = ptr - refobj_offset;
    ref = (struct ref_obj *)ptr;

    if (!data || !ref || (ref->magic != REFOBJ_MAGIC)) {
        return (ret);
    }

    /*double check just incase im gone*/
    if (!pthread_mutex_lock(ref->lock)) {
        if ((ref->magic == REFOBJ_MAGIC) && (ref->cnt > 0))
       {
            ref->cnt++;
            ret = ref->cnt;
        }
        pthread_mutex_unlock(ref->lock);
    }

    return (ret);
}
```

5.10.3.13 int objsize (void * *data*)

Definition at line 191 of file refobj.c.

References ref_obj::data, ref_obj::lock, ref_obj::magic, REFOBJ_MAGIC, refobj_offset, and ref_obj::size.

Referenced by ldap_encattr().

```
                                {
    char *ptr = data;
    int ret = 0;
    struct ref_obj *ref;

    if (!data) {
        return (ret);
    }

    ptr = ptr - refobj_offset;
    ref = (struct ref_obj *)ptr;

    if (ref->magic == REFOBJ_MAGIC) {
        pthread_mutex_lock(ref->lock);
        ret = ref->size;
        pthread_mutex_unlock(ref->lock);
    }
    return (ret);
}
```

5.10.3.14 int objtrylock (void * *data*)

Definition at line 224 of file refobj.c.

References ref_obj::data, ref_obj::lock, ref_obj::magic, REFOBJ_MAGIC, and refobj_offset.

```
                                    {
    char *ptr = data;
    struct ref_obj *ref;
```

```
    ptr = ptr - refobj_offset;
    ref = (struct ref_obj *)ptr;

    if (ref->magic == REFOBJ_MAGIC) {
        return ((pthread_mutex_trylock(ref->lock)) ? -1 : 0);
    }
    return (-1);
}
```

5.10.3.15 int objunlock (void * *data*)

Definition at line 237 of file refobj.c.

References ref_obj::data, ref_obj::lock, ref_obj::magic, REFOBJ_MAGIC, and refobj_offset.

Referenced by addtobucket(), bucket_list_cnt(), create_kernmac(), create_kernvlan(), curl_postitem(), curlinit(), dtls_listenssl(), dtlshandltimeout(), dtlstimeout(), dts_ldapsearch(), dtsl_serveropts(), framework_mkthread(), get_iface_index(), ldap_attrvals(), ldap_count(), ldap_doadd(), ldap_domodify(), ldap_getattribute(), ldap_getdn(), ldap_getent(), ldap_saslbind(), ldap_simplebind(), nf_ctrack_delete(), nf_ctrack_dump(), nf_ctrack_nat(), nfqueue_attach(), remove_bucket_loop(), set_interface_addr(), set_interface_flags(), set_interface_ipaddr(), set_interface_name(), socketread_d(), socketserver(), socketwrite_d(), ssl_shutdown(), url_escape(), url_unescape(), xml_addnode(), xml_appendnode(), xml_createpath(), xml_delete(), xml_doctobuffer(), xml_getfirstnode(), xml_gethash(), xml_getnextnode(), xml_getrootnode(), xml_modify(), xml_savefile(), xml_setattr(), xml_unlink(), xml_xpath(), xslt_addparam(), xslt_apply(), xslt_apply_buffer(), and xslt_clearparam().

```
                                {
    char *ptr = data;
    struct ref_obj *ref;

    ptr = ptr - refobj_offset;
    ref = (struct ref_obj *)ptr;

    if (ref->magic == REFOBJ_MAGIC) {
        pthread_mutex_unlock(ref->lock);
    }
    return (0);
}
```

5.10.3.16 int objunref (void * *data*)

Definition at line 133 of file refobj.c.

References ref_obj::cnt, ref_obj::data, ref_obj::destroy, ref_obj::lock, ref_obj::magic, REFOBJ_MAGIC, refobj_offset, and ref_obj::size.

Referenced by add_modifyval(), add_radserver(), addtobucket(), attr2bl(), bucket_list_find_key(), bucketlist_callback(), close_socket(), closenetlink(), create_kernmac(), create_kernvlan(), curl_setauth_cb(), curl_setprogress(), curlclose(), curlinit(), dtls_listenssl(), dts_ldapsearch(), framework_init(), framework_mkcore(), framework_mkthread(), framework_unixsocket(), free_add(), free_attr(), free_attrval(), free_attrvalarr(), free_curlpassword(), free_entarr(), free_entry(), free_ldapconn(), free_modify(), free_modreq(), free_progress(), free_rdn(), free_rdnarr(), free_result(), free_xsltdoc(), get_category_loop(), get_category_next(), get_config_category(), get_config_file(), get_iface_index(), ifhwaddr(), ldap_add_attr(), ldap_addinit(), ldap_close(), ldap_connect(), ldap_doadd(), ldap_domodify(), ldap_getent(), ldap_mod_add(), ldap_mod_addattr(), ldap_mod_del(), ldap_mod_delattr(), ldap_mod_rep(), ldap_mod_repattr(), ldap_modifyinit(), ldap_rebind_proc(), ldap_saslbind(), ldap_simplebind(), ldap_simplerebind(), ldap_unref_attr(), ldap_unref_entry(), make_socket(), new_modreq(), nf_ctrack_close(), nf_ctrack_endtrace(), nf_ctrack_trace(), nfqueue_attach(), printgnu(), process_config(), remove_bucket_item(), remove_bucket_loop(), rfc6296_map_add(), rfc6296_test(), set_interface_addr(), set_interface_flags(), set_interface_ipaddr(), set_interface_name(), socketread_d(), socketwrite_d(), startthreads(), stop_bucket_loop(), unrefconfigfiles(), url_escape(), url_unescape(), xml_addnode(), xml_appendnode(), xml_close(), xml_createpath(), xml_getattr(), xml_getfirstnode(), xml_gethash(), xml_getnextnode(), xml_loadbuf(), xml_loaddoc(), xml_nodetohash(), xml_setnodes(), xml_xpath(), xslt_addparam(), xslt_apply(), xslt_apply_buffer(), xslt_clearparam(), and xslt_close().

```
                                {
    char *ptr = data;
    struct ref_obj *ref;
    int ret = -1;
    pthread_mutex_t *lock;

    if (!data) {
        return (ret);
    }

    ptr = ptr - refobj_offset;
    ref = (struct ref_obj *)ptr;

    if ((ref->magic == REFOBJ_MAGIC) && (ref->cnt)) {
        pthread_mutex_lock(ref->lock);
        ref->cnt--;
        ret = ref->cnt;
        /* free the object its no longer in use*/
        if (!ret) {
            lock = ref->lock;
            ref->lock = NULL;
            ref->magic = 0;
            ref->size = 0;
            ref->data = NULL;
            if (ref->destroy) {
                ref->destroy(data);
            }
            pthread_mutex_unlock(lock);
            pthread_mutex_destroy(lock);
            free(lock);
            free(ref);
        } else {
            pthread_mutex_unlock(ref->lock);
        }
    }
    return (ret);
}
```

5.10.3.17 void remove_bucket_item (struct bucket_list * *blist*, void * *data*)

Definition at line 500 of file refobj.c.

References bucket_list::bucketbits, ref_obj::data, blist_obj::data, blist_obj::hash, bucket_list::list, bucket_list::locks, blist_obj::next, objunref(), and blist_obj::prev.

Referenced by ldap_unref_attr(), and ldap_unref_entry().

```
                                                                    {
    struct blist_obj *entry;
    int hash, bucket;

    hash = gethash(blist, data, 0);
    bucket = ((hash >> (32 - blist->bucketbits)) & ((1 << blist->
      bucketbits) - 1));

    pthread_mutex_lock(&blist->locks[bucket]);
    entry = blist_gotohash(blist->list[bucket], hash + 1, blist->bucketbits
      );
    if (entry && entry->hash == hash) {
        if (entry->next && (entry == blist->list[bucket])) {
            entry->next->prev = entry->prev;
            blist->list[bucket] = entry->next;
        } else
            if (entry->next) {
                entry->next->prev = entry->prev;
                entry->prev->next = entry->next;
            } else
                if (entry == blist->list[bucket]) {
                    blist->list[bucket] = NULL;
                } else {
                    entry->prev->next = NULL;
                    blist->list[bucket]->prev = entry->prev;
                }
        objunref(entry->data->data);
        free(entry);
    }
    pthread_mutex_unlock(&blist->locks[bucket]);
}
```

5.10.3.18 void remove_bucket_loop (struct bucket_loop * *bloop*)

Definition at line 533 of file refobj.c.

References bucket_loop::blist, bucket_loop::bucket, bucket_list::bucketbits, bucket_list::count, bucket_loop::cur, bucket_loop::cur_hash, ref_obj::data, blist_obj::data, blist_obj::hash, bucket_list::list, bucket_list::locks, blist_obj::next, objlock(), objunlock(), objunref(), blist_obj::prev, bucket_list::version, and bucket_loop::version.

```
                                                                {
    struct bucket_list *blist = bloop->blist;
    int bucket = bloop->bucket;

    pthread_mutex_lock(&blist->locks[bloop->bucket]);
    /*if the bucket has altered need to verify i can remove*/
    if (bloop->cur_hash && (!bloop->cur || (blist->version[
      bloop->bucket] != bloop->version))) {
        bloop->cur = blist_gotohash(blist->list[bloop->bucket],
      bloop->cur_hash + 1, blist->bucketbits);
        if (!bloop->cur || (bloop->cur->hash != bloop->cur_hash
      )) {
            pthread_mutex_unlock(&blist->locks[bucket]);
            return;
        }
    }

    if (!bloop->cur) {
        pthread_mutex_unlock(&blist->locks[bucket]);
        return;
    }

    if (bloop->cur->next && (bloop->cur == blist->list[bucket]))
      {
        bloop->cur->next->prev = bloop->cur->prev;
        blist->list[bucket] = bloop->cur->next;
    } else
        if (bloop->cur->next) {
            bloop->cur->next->prev = bloop->cur->prev;
            bloop->cur->prev->next = bloop->cur->next;
        } else
            if (bloop->cur == blist->list[bucket]) {
                blist->list[bucket] = NULL;
            } else {
                bloop->cur->prev->next = NULL;
                blist->list[bucket]->prev = bloop->cur->prev;
            }

    objunref(bloop->cur->data->data);
    free(bloop->cur);
    bloop->cur_hash = 0;
    bloop->cur = NULL;
    blist->version[bucket]++;
    bloop->version++;
    pthread_mutex_unlock(&blist->locks[bucket]);

    objlock(blist);
    blist->count--;
    objunlock(blist);
}
```

5.10.3.19 void stop_bucket_loop (struct bucket_loop * *bloop*)

Definition at line 449 of file refobj.c.

References bucket_loop::blist, and objunref().

Referenced by bucketlist_callback(), ldap_doadd(), and ldap_domodify().

```
                                                        {

    if (bloop) {
        objunref(bloop->blist);
        objunref(bloop);
    }
}
```

5.11 Posix thread interface

Functions for starting and managing threads.

Files

- file thread.c

  Functions for starting and managing threads.

Data Structures

- struct thread_pvt

  thread struct used to create threads data needs to be first element
- struct threadcontainer

  Global threads data.

Macros

- #define SIGHUP 1

  Define SIGHUP as 1 if its not defined.
- #define THREAD_MAGIC 0xfeedf158

  32 bit magic value to help determine thread is ok

Typedefs

- typedef void(∗ threadcleanup)(void ∗)

  Function called after thread termination.
- typedef void ∗(∗ threadfunc)(void ∗∗)

  Thread function.
- typedef int(∗ threadsighandler)(int, void ∗)

  Thread signal handler function.

Enumerations

- enum threadopt { TL_THREAD_NONE = 1 << 0, TL_THREAD_RUN = 1 << 1, TL_THREAD_DONE = 1 << 2 }

  Thread status a thread can be disabled by unsetting TL_THREAD_RUN.

Functions

- int framework_threadok (void ∗data)

  let threads check there status by passing in a pointer to there data
- int startthreads (void)

  initialise the threadlist start manager thread
- void stopthreads (void)

  Stoping the manager thread will stop all other threads.
- struct thread_pvt ∗ framework_mkthread (threadfunc func, threadcleanup cleanup, threadsighandler sig_handler, void ∗data)

  create a thread result must be unreferenced
- void jointhreads (void)

  Join the manager thread.

Variables

- struct threadcontainer * threads = NULL

  Thread control data.

5.11.1 Detailed Description

Functions for starting and managing threads. The thread interface consists of a management thread managing a hashed bucket list of threads running optional clean up when done.

5.11.2 Macro Definition Documentation

5.11.2.1 #define SIGHUP 1

Define SIGHUP as 1 if its not defined.

Definition at line 40 of file thread.c.

5.11.2.2 #define THREAD_MAGIC 0xfeedf158

32 bit magic value to help determine thread is ok

Definition at line 46 of file thread.c.

Referenced by framework_mkthread(), and framework_threadok().

5.11.3 Typedef Documentation

5.11.3.1 typedef void(* threadcleanup)(void *)

Function called after thread termination.

See Also

framework_mkthread()

Parameters

data	Reference of thread data.

Definition at line 157 of file dtsapp.h.

5.11.3.2 typedef void*(* threadfunc)(void **)

Thread function.

See Also

framework_mkthread()

Parameters

data	Poinnter to reference of thread data.

Definition at line 163 of file dtsapp.h.

5.11.3.3 typedef int(* threadsighandler)(int, void *)

Thread signal handler function.

See Also

framework_mkthread()

Parameters

data	Reference of thread data.

Definition at line 169 of file dtsapp.h.

5.11.4 Enumeration Type Documentation

5.11.4.1 enum threadopt

Thread status a thread can be disabled by unsetting TL_THREAD_RUN.

Enumerator:

TL_THREAD_NONE No status.

TL_THREAD_RUN thread is marked as running

TL_THREAD_DONE thread is marked as complete

Definition at line 49 of file thread.c.

```
            {
    TL_THREAD_NONE   = 1 << 0,
    TL_THREAD_RUN    = 1 << 1,
    TL_THREAD_DONE   = 1 << 2
};
```

5.11.5 Function Documentation

5.11.5.1 struct thread_pvt* framework_mkthread (threadfunc *func*, threadcleanup *cleanup*, threadsighandler *sig_handler*, void * *data*) [read]

create a thread result must be unreferenced

Parameters

func	Function to run thread on.
cleanup	Cleanup function to run.
sig_handler	Thread signal handler.
data	Data to pass to callbacks.

Returns

a thread structure that must be un referencend.

Definition at line 257 of file thread.c.

References addtobucket(), thread_pvt::cleanup, thread_pvt::data, thread_pvt::flags, thread_pvt::func, threadcontainer::list, thread_pvt::magic, threadcontainer::manager, objalloc(), objlock(), objref(), objunlock(), objunref(), thread_pvt::sighandler, thread_pvt::thr, and THREAD_MAGIC.

Referenced by framework_unixsocket(), nf_ctrack_trace(), radconnect(), and startthreads().

```
                                                    {
    struct thread_pvt *thread;

    /* dont allow threads if no manager or it not started*/
    if ((!threads || !threads->manager) && (func !=
      managethread)) {
        return NULL;
    }

    if (!(thread = objalloc(sizeof(*thread), NULL))) {
        return NULL;
    }

    thread->data = data;
    thread->flags = 0;
    thread->cleanup = cleanup;
    thread->sighandler = sig_handler;
    thread->func = func;
    thread->magic = THREAD_MAGIC;

    /* grab a ref to data for thread to make sure it does not go away*/
    objref(thread->data);
    if (pthread_create(&thread->thr, NULL, threadwrap, thread)) {
        objunref(thread->data);
        objunref(thread);
        return NULL;
    }

    /* am i up and running move ref to list*/
    if (!pthread_kill(thread->thr, 0)) {
        if (threads && threads->list) {
            objlock(threads);
            addtobucket(threads->list, thread);
            objunlock(threads);
            return (thread);
        } else {
            objunref(thread->data);
            objunref(thread);
        }
    } else {
        objunref(thread->data);
        objunref(thread);
    }

    return NULL;
}
```

5.11.5.2 int framework_threadok (void * *data*)

let threads check there status by passing in a pointer to there data

Parameters

data	Reference to thread data

Returns

0 if the thread should terminate.

Definition at line 115 of file thread.c.

References thread_pvt::data, thread_pvt::magic, testflag, thread_pvt::thr, THREAD_MAGIC, and TL_THREAD_RUN.

```
                                            {
    struct thread_pvt *thr = data;

    if (thr && (thr->magic == THREAD_MAGIC)) {
        return (testflag(thr, TL_THREAD_RUN));
    }
    return (0);
}
```

5.11.5.3 void jointhreads (void)

Join the manager thread.

This will be done when you have issued stopthreads and are waiting for threads to exit.

Definition at line 307 of file thread.c.

References threadcontainer::manager, and thread_pvt::thr.

Referenced by framework_init().

```
                                {
    if (threads && threads->manager) {
        pthread_join(threads->manager->thr, NULL);
    }
}
```

5.11.5.4 int startthreads (void)

initialise the threadlist start manager thread

Returns

1 On success 0 on failure.

Definition at line 203 of file thread.c.

References create_bucketlist(), framework_mkthread(), threadcontainer::list, threadcontainer::manager, objalloc(), and objunref().

Referenced by framework_init().

```
                                {
    if (!threads && !(threads = objalloc(sizeof(*threads
      ), close_threads))) {
        return (0);
    }

    if (!threads->list && !(threads->list =
      create_bucketlist(4, hash_thread))) {
        objunref(threads);
        return (0);
    }

    if (!threads->manager && !(threads->manager =
      framework_mkthread(managethread, NULL, manager_sig, NULL))) {
        objunref(threads);
        return (0);
    }

    return (1);
}
```

5.11.5.5 void stopthreads (void)

Stoping the manager thread will stop all other threads.

Definition at line 222 of file thread.c.

References clearflag, threadcontainer::manager, and TL_THREAD_RUN.

Referenced by framework_init().

```
                                {
    if (threads) {
        clearflag(threads->manager, TL_THREAD_RUN
      );
    }
}
```

5.11.6 Variable Documentation

5.11.6.1 struct threadcontainer* threads = NULL

Thread control data.

Definition at line 89 of file thread.c.

5.12 Unix socket thread

Attach a thread to a unix socket calling a callback on connect.

Files

- file unixsock.c

  Attach a thread to a unix socket calling a callback on connect.

Data Structures

- struct framework_sockthread

  Unix socket data structure.

Functions

- void framework_unixsocket (char *sock, int protocol, int mask, threadfunc connectfunc, threadcleanup cleanup)

  Create and run UNIX socket thread.

5.12.1 Detailed Description

Attach a thread to a unix socket calling a callback on connect. A thread is started on the sockect and will start a new client thread on each connection with the socket as the data

5.12.2 Function Documentation

5.12.2.1 void framework_unixsocket (char * *sock*, int *protocol*, int *mask*, threadfunc *connectfunc*, threadcleanup *cleanup*)

Create and run UNIX socket thread.

Parameters

sock	Path to UNIX socket.
protocol	Protocol number.
mask	Umask for the socket.
connectfunc	Thread to start on connect.
cleanup	Thread cleanup callback.

Definition at line 159 of file unixsock.c.

References framework_sockthread::cleanup, framework_sockthread::client, framework_mkthread(), framework_sockthread::mask, objalloc(), objunref(), framework_sockthread::protocol, and framework_sockthread::sock.

```
                                                {
struct framework_sockthread *unsock;
void *thread;

unsock = objalloc(sizeof(*unsock), NULL);
strncpy(unsock->sock, sock, UNIX_PATH_MAX);
unsock->mask = mask;
unsock->client = connectfunc;
unsock->cleanup = cleanup;
unsock->protocol = protocol;
thread = framework_mkthread(unsock_serv, NULL, NULL,
  unsock);
```

```
    objunref(thread);
}
```

5.13 Micelaneous utilities.

Utilities commonly used.

Files

- file util.c

  Utilities commonly used.

Functions

- void seedrand (void)

  Seed openssl random number generator.
- int genrand (void *buf, int len)

  Generate random sequence.
- void sha512sum2 (unsigned char *buff, const void *data, unsigned long len, const void *data2, unsigned long len2)

  Calculate the SHA2-512 hash accross 2 data chunks.
- void sha512sum (unsigned char *buff, const void *data, unsigned long len)

  Calculate the SHA2-512 hash.
- void sha256sum2 (unsigned char *buff, const void *data, unsigned long len, const void *data2, unsigned long len2)

  Calculate the SHA2-256 hash accross 2 data chunks.
- void sha256sum (unsigned char *buff, const void *data, unsigned long len)

  Calculate the SHA2-256 hash.
- void sha1sum2 (unsigned char *buff, const void *data, unsigned long len, const void *data2, unsigned long len2)

  Calculate the SHA1 hash accross 2 data chunks.
- void sha1sum (unsigned char *buff, const void *data, unsigned long len)

  Calculate the SHA1 hash.
- void md5sum2 (unsigned char *buff, const void *data, unsigned long len, const void *data2, unsigned long len2)

  Calculate the MD5 hash accross 2 data chunks.
- void md5sum (unsigned char *buff, const void *data, unsigned long len)

  Calculate the MD5 hash.
- int md5cmp (unsigned char *digest1, unsigned char *digest2)

  Compare two md5 hashes.
- int sha1cmp (unsigned char *digest1, unsigned char *digest2)

  Compare two SHA1 hashes.
- int sha256cmp (unsigned char *digest1, unsigned char *digest2)

  Compare two SHA2-256 hashes.
- int sha512cmp (unsigned char *digest1, unsigned char *digest2)

  Compare two SHA2-512 hashes.
- void md5hmac (unsigned char *buff, const void *data, unsigned long len, const void *key, unsigned long klen)

  Hash Message Authentication Codes (HMAC) MD5.
- void sha1hmac (unsigned char *buff, const void *data, unsigned long len, const void *key, unsigned long klen)

  Hash Message Authentication Codes (HMAC) SHA1.
- void sha256hmac (unsigned char *buff, const void *data, unsigned long len, const void *key, unsigned long klen)

Hash Message Authentication Codes (HMAC) SHA2-256.

- void sha512hmac (unsigned char *buff, const void *data, unsigned long len, const void *key, unsigned long klen)

  Hash Message Authentication Codes (HMAC) SHA2-512.
- int strlenzero (const char *str)

  Check if a string is zero length.
- char * ltrim (char *str)

  Trim white space at the begining of a string.
- char * rtrim (const char *str)

  Trim white space at the end of a string.
- char * trim (const char *str)

  Trim whitesapce from the beggining and end of a string.
- uint64_t tvtontp64 (struct timeval *tv)

  Convert a timeval struct to 64bit NTP time.
- uint16_t checksum (const void *data, int len)

  Obtain the checksum for a buffer.
- uint16_t checksum_add (const uint16_t checksum, const void *data, int len)

  Obtain the checksum for a buffer adding a checksum.
- uint16_t verifysum (const void *data, int len, const uint16_t check)

  Verify a checksum.
- void touch (const char *filename, uid_t user, gid_t group)

  Create a file and set user and group.
- char * b64enc_buf (const char *message, uint32_t len, int nonl)

  Base 64 encode a buffer.
- char * b64enc (const char *message, int nonl)

  Base 64 encode a string.

5.13.1 Detailed Description

Utilities commonly used.

5.13.2 Function Documentation

5.13.2.1 char* b64enc (const char * *message*, int *nonl*)

Base 64 encode a string.

Parameters

message	String to encode.
nonl	Encode the data all on one line if non zero.

Returns

Reference to base64 encoded string.

Definition at line 513 of file util.c.

References b64enc_buf().

```
                                                {
    return b64enc_buf(message, strlen(message), nonl);
}
```

5.13.2.2 char* b64enc_buf (const char * *message*, uint32_t *len*, int *nonl*)

Base 64 encode a buffer.

Parameters

message	Buffer to encode.
len	Length of the buffer.
nonl	Encode the data all on one line if non zero.

Returns

Reference to base64 encoded string.

Definition at line 480 of file util.c.

References objalloc().

Referenced by b64enc(), and ldap_encattr().

```
{
    BIO *bmem, *b64;
    BUF_MEM *ptr;
    char *buffer;
    double encodedSize;

    encodedSize = 1.36*len;
    buffer = objalloc(encodedSize+1, NULL);

    b64 = BIO_new(BIO_f_base64());
    bmem = BIO_new(BIO_s_mem());
    b64 = BIO_push(b64, bmem);
    if (nonl) {
        BIO_set_flags(b64, BIO_FLAGS_BASE64_NO_NL);
    }
    BIO_write(b64, message, len);
    BIO_flush(b64);
    BIO_get_mem_ptr(b64, &ptr);

    buffer = objalloc(ptr->length+1, NULL);
    memcpy(buffer, ptr->data, ptr->length);


    BIO_free_all(b64);

    return buffer;
}
```

5.13.2.3 uint16_t checksum (const void * *data*, int *len*)

Obtain the checksum for a buffer.

Parameters

data	Buffer to create checksum of.
len	Buffer length.

Returns

Chechsum of data.

Definition at line 426 of file util.c.

Referenced by icmpchecksum(), ipv4checksum(), ipv4tcpchecksum(), ipv4udpchecksum(), and rfc6296_map_-add().

```
{
    return (_checksum(data, len, 0));
}
```

5.13.2.4 uint16_t checksum_add (const uint16_t *checksum*, const void * *data*, int *len*)

Obtain the checksum for a buffer adding a checksum.

Parameters

checksum	Checksum to add to generated checksum.
data	Buffer to create checksum of.
len	Buffer length.

Returns

Chechsum of data.

Definition at line 437 of file util.c.

Referenced by ipv4tcpchecksum(), and ipv4udpchecksum().

```
    {
    return (_checksum(data, len, ~checksum));
}
```

5.13.2.5 int genrand (void * *buf*, int *len*)

Generate random sequence.

Parameters

buf	Buffer to write random data.
len	Length to write.

Returns

1 on success 0 otherwise.

Definition at line 80 of file util.c.

Referenced by new_radpacket(), radconnect(), randhwaddr(), and sslstartup().

```
                                    {
    return (RAND_bytes(buf, len));
}
```

5.13.2.6 char* ltrim (char * *str*)

Trim white space at the begining of a string.

Parameters

str	String to trim.

Returns

Pointer to trimed string.

Definition at line 327 of file util.c.

References strlenzero().

Referenced by trim().

```
                                {
    char *cur = str;

    if (strlenzero(str)) {
        return (str);
    }

    while(isspace(cur[0])) {
        cur++;
    }

    return (cur);
}
```

5.13.2.7 int md5cmp (unsigned char * *digest1*, unsigned char * *digest2*)

Compare two md5 hashes.

Parameters

digest1	Digest to compare.
digest2	Digest to compare.

Returns

0 on equality.

Definition at line 209 of file util.c.

```
                                                                {
    return (_digest_cmp(digest1, digest2, 16));
}
```

5.13.2.8 void md5hmac (unsigned char * *buff*, const void * *data*, unsigned long *len*, const void * *key*, unsigned long *klen*)

Hash Message Authentication Codes (HMAC) MD5.

Parameters

buff	HMAC returned in this buffer (16 bytes).
data	Data to sign.
len	Length of data.
key	Key to signwith.
klen	Length of key.

Definition at line 272 of file util.c.

References md5sum2().

```
                                                {
    _hmac(buff, data, len, key, klen, md5sum2, 16);
}
```

5.13.2.9 void md5sum (unsigned char * *buff*, const void * *data*, unsigned long *len*)

Calculate the MD5 hash.

Parameters

buff	buffer to place the hash (16 bytes).
data	First data chunk to calculate.
len	Length of data.

Definition at line 189 of file util.c.

References md5sum2().

```
                                                                    {
    md5sum2(buff, data, len, NULL, 0);
}
```

5.13.2.10 void md5sum2 (unsigned char * *buff*, const void * *data*, unsigned long *len*, const void * *data2*, unsigned long *len2*)

Calculate the MD5 hash accross 2 data chunks.

Parameters

buff	buffer to place the hash (16 bytes).
data	First data chunk to calculate.
len	Length of data.
data2	Second data chunk to calculate.
len2	Length of data2.

Definition at line 173 of file util.c.

Referenced by md5hmac(), and md5sum().

```
                                                                    {
    MD5_CTX c;

    MD5_Init(&c);
    MD5_Update(&c, data, len);
    if (data2) {
        MD5_Update(&c, data2, len2);
    }
    MD5_Final(buff, &c);
}
```

5.13.2.11 char* rtrim (const char * *str*)

Trim white space at the end of a string.

Parameters

str	String to trim.

Returns

Pointer to trimed string.

Definition at line 346 of file util.c.

References strlenzero().

Referenced by trim().

```
                                                {
```

```
    int len;
    char *cur = (char *)str;

    if (strlenzero(str)) {
        return (cur);
    }

    len = strlen(str) - 1;
    while(len && isspace(cur[len])) {
        cur[len] = '\0';
        len--;
    }

    return (cur);
}
```

5.13.2.12 void seedrand (void)

Seed openssl random number generator.

This should be run at application startup

Todo This wont work on WIN32

Definition at line 66 of file util.c.

Referenced by framework_init().

```
                        {
    int fd = open("/dev/random", O_RDONLY);
    int len;
    char    buf[64];

    len = read(fd, buf, 64);
    RAND_seed(buf, len);
}
```

5.13.2.13 int sha1cmp (unsigned char * *digest1*, unsigned char * *digest2*)

Compare two SHA1 hashes.

Parameters

digest1	Digest to compare.
digest2	Digest to compare.

Returns

0 on equality.

Definition at line 218 of file util.c.

```
                                                                {
    return (_digest_cmp(digest1, digest2, 20));
}
```

5.13.2.14 void sha1hmac (unsigned char * *buff*, const void * *data*, unsigned long *len*, const void * *key*, unsigned long *klen*)

Hash Message Authentication Codes (HMAC) SHA1.

Parameters

buff	HMAC returned in this buffer (20 bytes).
data	Data to sign.
len	Length of data.
key	Key to signwith.
klen	Length of key.

Definition at line 283 of file util.c.

References sha1sum2().

```
                                                {
    _hmac(buff, data, len, key, klen, sha1sum2, 20);
}
```

5.13.2.15 void sha1sum (unsigned char * *buff*, const void * *data*, unsigned long *len*)

Calculate the SHA1 hash.

Parameters

buff	buffer to place the hash (20 bytes).
data	First data chunk to calculate.
len	Length of data.

Definition at line 162 of file util.c.

References sha1sum2().

```
                                                                    {
    sha1sum2(buff, data, len, NULL, 0);
}
```

5.13.2.16 void sha1sum2 (unsigned char * *buff*, const void * *data*, unsigned long *len*, const void * *data2*, unsigned long *len2*)

Calculate the SHA1 hash accross 2 data chunks.

Parameters

buff	buffer to place the hash (20 bytes).
data	First data chunk to calculate.
len	Length of data.
data2	Second data chunk to calculate.
len2	Length of data2.

Definition at line 146 of file util.c.

Referenced by get_ip6_addrprefix(), sha1hmac(), and sha1sum().

```
                                                {
    SHA_CTX c;

    SHA_Init(&c);
    SHA_Update(&c, data, len);
    if (data2) {
        SHA_Update(&c, data2, len2);
    }
    SHA_Final(buff, &c);
}
```

5.13.2.17 int sha256cmp (unsigned char * *digest1*, unsigned char * *digest2*)

Compare two SHA2-256 hashes.

Parameters

digest1	Digest to compare.
digest2	Digest to compare.

Returns

0 on equality.

Definition at line 227 of file util.c.

```
                                                                {
    return (_digest_cmp(digest1, digest2, 32));
}
```

5.13.2.18 void sha256hmac (unsigned char * *buff*, const void * *data*, unsigned long *len*, const void * *key*, unsigned long *klen*)

Hash Message Authentication Codes (HMAC) SHA2-256.

Parameters

buff	HMAC returned in this buffer (32 bytes).
data	Data to sign.
len	Length of data.
key	Key to signwith.
klen	Length of key.

Definition at line 294 of file util.c.

References sha256sum2().

```
                                                    {
    _hmac(buff, data, len, key, klen, sha256sum2, 32);
}
```

5.13.2.19 void sha256sum (unsigned char * *buff*, const void * *data*, unsigned long *len*)

Calculate the SHA2-256 hash.

Parameters

buff	buffer to place the hash (32 bytes).
data	First data chunk to calculate.
len	Length of data.

Definition at line 135 of file util.c.

References sha256sum2().

```
      {
    sha256sum2(buff, data, len, NULL, 0);
}
```

5.13.2.20 void sha256sum2 (unsigned char * *buff*, const void * *data*, unsigned long *len*, const void * *data2*, unsigned long *len2*)

Calculate the SHA2-256 hash accross 2 data chunks.

Parameters

buff	buffer to place the hash (32 bytes).
data	First data chunk to calculate.
len	Length of data.
data2	Second data chunk to calculate.
len2	Length of data2.

Definition at line 119 of file util.c.

Referenced by sha256hmac(), and sha256sum().

```
                                                {
    SHA256_CTX c;

    SHA256_Init(&c);
    SHA256_Update(&c, data, len);
    if (data2) {
        SHA256_Update(&c, data2, len2);
    }
    SHA256_Final(buff, &c);
}
```

5.13.2.21 int sha512cmp (unsigned char * *digest1*, unsigned char * *digest2*)

Compare two SHA2-512 hashes.

Parameters

digest1	Digest to compare.
digest2	Digest to compare.

Returns

0 on equality.

Definition at line 236 of file util.c.

```
                                                        {
    return (_digest_cmp(digest1, digest2, 64));
}
```

5.13.2.22 void sha512hmac (unsigned char * *buff*, const void * *data*, unsigned long *len*, const void * *key*, unsigned long *klen*)

Hash Message Authentication Codes (HMAC) SHA2-512.

Parameters

buff	HMAC returned in this buffer (64 bytes).
data	Data to sign.
len	Length of data.
key	Key to signwith.
klen	Length of key.

Definition at line 305 of file util.c.

References sha512sum2().

```
                                                {
    _hmac(buff, data, len, key, klen, sha512sum2, 64);
}
```

5.13.2.23 void sha512sum (unsigned char * *buff*, const void * *data*, unsigned long *len*)

Calculate the SHA2-512 hash.

Parameters

buff	buffer to place the hash (64 bytes).
data	First data chunk to calculate.
len	Length of data.

Definition at line 107 of file util.c.

References sha512sum2().

```
      {
    sha512sum2(buff, data, len, NULL, 0);
}
```

5.13.2.24 void sha512sum2 (unsigned char * *buff*, const void * *data*, unsigned long *len*, const void * *data2*, unsigned long *len2*)

Calculate the SHA2-512 hash accross 2 data chunks.

Parameters

buff	buffer to place the hash (64 bytes).
data	First data chunk to calculate.
len	Length of data.
data2	Second data chunk to calculate.
len2	Length of data2.

Definition at line 91 of file util.c.

Referenced by sha512hmac(), and sha512sum().

```
                                                {
    SHA512_CTX c;

    SHA512_Init(&c);
    SHA512_Update(&c, data, len);
    if (data2) {
        SHA512_Update(&c, data2, len2);
    }
    SHA512_Final(buff, &c);
}
```

5.13.2.25 int strlenzero (const char * *str*)

Check if a string is zero length.

strlen can not be used on a NULL string this is a quick and dirty util to check it.

Parameters

str	String to check.

Returns

1 if the string is null or zero length

Definition at line 315 of file util.c.

Referenced by create_kernmac(), create_kernvlan(), get_category_next(), ifhwaddr(), ltrim(), process_config(), and rtrim().

```
                                        {
    if (str && strlen(str)) {
        return (0);
    }
    return (1);
}
```

5.13.2.26 void touch (const char * *filename*, uid_t *user*, gid_t *group*)

Create a file and set user and group.

Todo WIN32 does not use uid/gid and move to file utils module.

Parameters

filename	File to create.
user	User ID to set ownership.
group	Group ID to set ownership.

Definition at line 458 of file util.c.

References touch().

Referenced by touch(), and xslt_apply().

```
                                                                {
    int res;
#else
extern void touch(const char *filename) {
#endif
    int fd;

    fd = creat(filename, 0600);
    close(fd);
#ifndef __WIN32__
    res = chown(filename, user, group);
    res++;
#endif
    return;
}
```

5.13.2.27 char* trim (const char * *str*)

Trim whitesapce from the beggining and end of a string.

Parameters

str	String to trim.

Returns

Trimed string.

Definition at line 367 of file util.c.

References ltrim(), and rtrim().

Referenced by process_config().

```
                                    {
    char *cur = (char *)str;

    cur = ltrim(cur);
    cur = rtrim(cur);
    return (cur);
}
```

5.13.2.28 uint64_t tvtontp64 (struct timeval * *tv*)

Convert a timeval struct to 64bit NTP time.

Parameters

tv	Timeval struct to convert.

Returns

64 bit NTP time value.

Definition at line 379 of file util.c.

Referenced by get_ip6_addrprefix().

```
                                          {
    return ((((uint64_t)tv->tv_sec + 2208988800u) << 32) + ((uint32_t)tv->
      tv_usec * 4294.967296));
}
```

5.13.2.29 uint16_t verifysum (const void * *data,* int *len,* const uint16_t *check*)

Verify a checksum.

Parameters

data	Data to generate checksum.
len	Length of data.
check	Checksum to check against.

Returns

0 when checksum is verified.

Definition at line 447 of file util.c.

```
                                                                    {
    return (_checksum(data, len, check));
}
```

5.14 Zlib Interface

Simplified interface to Compress/Uncompress/Test a buffer.

Files

- file zlib.c

  Simplified interface to Compress/Uncompress/Test a buffer.

Data Structures

- struct zobj

  Zlib buffer used for compression and decompression.

Functions

- struct zobj * zcompress (uint8_t *buff, uint16_t len, uint8_t level)

  Allocate a buffer and return it with compressed data.
- void zuncompress (struct zobj *buff, uint8_t *obuff)

  Uncompress zobj buffer to buffer.
- int is_gzip (uint8_t *buf, int buf_size)

  check a buffer if it contains gzip magic
- uint8_t * gzinflatebuf (uint8_t *buf_in, int buf_size, uint32_t *len)

  Ungzip a buffer.

5.14.1 Detailed Description

Simplified interface to Compress/Uncompress/Test a buffer.

5.14.2 Function Documentation

5.14.2.1 uint8_t* gzinflatebuf (uint8_t * *buf_in*, int *buf_size*, uint32_t * *len*)

Ungzip a buffer.

Parameters

buf_in	Buffer to inflate.
buf_size	Size of buf_in buffer.
len	Pointer that will contain the uncompressed data length.

Returns

Uncompressed data in a buffer or NULL on error.

Definition at line 104 of file zlib.c.

References realloc.

Referenced by curl_ungzip().

```
{
    z_stream zdat;
    uint8_t *buf = NULL, *tmp;
```

```
    int res;

    zdat.opaque = NULL;
    zdat.zalloc = NULL;
    zdat.zfree = NULL;

    zdat.next_in = buf_in;
    zdat.avail_in = buf_size;
    zdat.next_out = buf;
    zdat.avail_out = 0;
    zdat.total_out = 0;

    if (inflateInit2(&zdat, 31)) {
        return NULL;
    }

    do {
        if (!(tmp = realloc(buf,zdat.total_out + (zdat.avail_in * 5) + 1
      ))) {
            res = Z_MEM_ERROR;
            break;
        } else {
            buf = tmp;
        }
        buf[zdat.total_out] = '\0';
        zdat.next_out = &buf[zdat.total_out];
        zdat.avail_out += zdat.avail_in * 5;
    } while ((res = inflate(&zdat, Z_NO_FLUSH)) == Z_OK);

    if (res == Z_STREAM_END) {
        buf = realloc(buf, zdat.total_out);
        *len = zdat.total_out;
    } else {
        free(buf);
        *len = 0;
        buf = NULL;
    }
    inflateEnd(&zdat);

    return buf;
}
```

5.14.2.2 int is_gzip (uint8_t * *buf*, int *buf_size*)

check a buffer if it contains gzip magic

Parameters

buf	buffer to check.
buf_size	buffer len it must be more than 4.

Returns

non zero value if the buffer contains gzip data

Definition at line 88 of file zlib.c.

Referenced by curl_ungzip().

```
                                                {
    if (buf_size < 4) {
        return 0;
    }
    if (memcmp(buf, gzipMagicBytes, 4)) {
        return 0;
    }
    return 1;
}
```

5.14.2.3 struct zobj* zcompress (uint8_t * *buff*, uint16_t *len*, uint8_t *level*) [read]

Allocate a buffer and return it with compressed data.

Parameters

buff	Buffer to compress.
len	Length of the buffer.
level	Compression level.

Returns

reference to zobj data structure containing compressed data or NULL on error.

Definition at line 50 of file zlib.c.

References zobj::buff, malloc, objalloc(), zobj::olen, and zobj::zlen.

```
                                                                    {
    struct zobj *ret;

    if (!(ret = objalloc(sizeof(*ret), zobj_free))) {
        return (NULL);
    }

    ret->zlen = compressBound(len);
    ret->olen = len;

    if (!(ret->buff = malloc(ret->zlen))) {
        return (NULL);
    }
    compress2(ret->buff, (uLongf *)&ret->zlen, buff, len, level);

    return (ret);
}
```

5.14.2.4 void zuncompress (struct zobj * *buff*, uint8_t * *obuff*)

Uncompress zobj buffer to buffer.

Parameters

buff	Compressed buffer to uncompress.
obuff	Buffer to uncompress too.

Warning

obuff needs to be large enough to contain the data.

Todo Implement this without needing original buff len using inflate

Definition at line 74 of file zlib.c.

References zobj::buff, zobj::olen, and zobj::zlen.

```
                                                        {
    uLongf olen = buff->olen;

    if (!obuff) {
        return;
    }

    uncompress(obuff, &olen, buff->buff, buff->zlen);
}
```

Chapter 6

Data Structure Documentation

6.1 basic_auth Struct Reference

`#include <dtsapp.h>`

Data Fields

- const char * user
- const char * passwd

6.1.1 Detailed Description

Definition at line 588 of file dtsapp.h.

6.1.2 Field Documentation

6.1.2.1 const char* basic_auth::passwd

Definition at line 590 of file dtsapp.h.

Referenced by curl_newauth().

6.1.2.2 const char* basic_auth::user

Definition at line 589 of file dtsapp.h.

Referenced by curl_newauth().

The documentation for this struct was generated from the following file:

- src/include/dtsapp.h

6.2 blist_obj Struct Reference

Data Fields

- int hash
- struct blist_obj * next

- struct blist_obj * prev
- struct ref_obj * data

6.2.1 Detailed Description

Definition at line 48 of file refobj.c.

6.2.2 Field Documentation

6.2.2.1 struct ref_obj* blist_obj::data

Definition at line 52 of file refobj.c.

Referenced by addtobucket(), bucket_list_find_key(), next_bucket_loop(), remove_bucket_item(), and remove_bucket_loop().

6.2.2.2 int blist_obj::hash

Definition at line 49 of file refobj.c.

Referenced by addtobucket(), bucket_list_find_key(), init_bucket_loop(), next_bucket_loop(), remove_bucket_item(), and remove_bucket_loop().

6.2.2.3 struct blist_obj* blist_obj::next

Definition at line 50 of file refobj.c.

Referenced by addtobucket(), next_bucket_loop(), remove_bucket_item(), and remove_bucket_loop().

6.2.2.4 struct blist_obj* blist_obj::prev

Definition at line 51 of file refobj.c.

Referenced by addtobucket(), next_bucket_loop(), remove_bucket_item(), and remove_bucket_loop().

The documentation for this struct was generated from the following file:

- src/refobj.c

6.3 bucket_list Struct Reference

Data Fields

- unsigned short bucketbits
- unsigned int count
- blisthash hash_func
- struct blist_obj ** list
- pthread_mutex_t * locks
- int * version

6.3.1 Detailed Description

Definition at line 56 of file refobj.c.

6.3.2 Field Documentation

6.3.2.1 unsigned short bucket_list::bucketbits

Definition at line 57 of file refobj.c.

Referenced by addtobucket(), bucket_list_find_key(), create_bucketlist(), next_bucket_loop(), remove_bucket_item(), and remove_bucket_loop().

6.3.2.2 unsigned int bucket_list::count

Definition at line 58 of file refobj.c.

Referenced by addtobucket(), bucket_list_cnt(), and remove_bucket_loop().

6.3.2.3 blisthash bucket_list::hash_func

Definition at line 59 of file refobj.c.

6.3.2.4 struct blist_obj** bucket_list::list

Definition at line 60 of file refobj.c.

Referenced by addtobucket(), bucket_list_find_key(), init_bucket_loop(), next_bucket_loop(), remove_bucket_item(), and remove_bucket_loop().

6.3.2.5 pthread_mutex_t* bucket_list::locks

Definition at line 61 of file refobj.c.

Referenced by addtobucket(), bucket_list_find_key(), init_bucket_loop(), next_bucket_loop(), remove_bucket_item(), and remove_bucket_loop().

6.3.2.6 int* bucket_list::version

Definition at line 62 of file refobj.c.

Referenced by addtobucket(), init_bucket_loop(), next_bucket_loop(), and remove_bucket_loop().

The documentation for this struct was generated from the following file:

- src/refobj.c

6.4 bucket_loop Struct Reference

Data Fields

- struct bucket_list * blist
- int bucket
- int version
- unsigned int head_hash
- unsigned int cur_hash
- struct blist_obj * head
- struct blist_obj * cur

6.4.1 Detailed Description

Definition at line 71 of file refobj.c.

6.4.2 Field Documentation

6.4.2.1 struct bucket_list* bucket_loop::blist

Definition at line 72 of file refobj.c.

Referenced by init_bucket_loop(), next_bucket_loop(), remove_bucket_loop(), and stop_bucket_loop().

6.4.2.2 int bucket_loop::bucket

Definition at line 73 of file refobj.c.

Referenced by init_bucket_loop(), next_bucket_loop(), and remove_bucket_loop().

6.4.2.3 struct blist_obj* bucket_loop::cur

Definition at line 78 of file refobj.c.

Referenced by next_bucket_loop(), and remove_bucket_loop().

6.4.2.4 unsigned int bucket_loop::cur_hash

Definition at line 76 of file refobj.c.

Referenced by next_bucket_loop(), and remove_bucket_loop().

6.4.2.5 struct blist_obj* bucket_loop::head

Definition at line 77 of file refobj.c.

Referenced by init_bucket_loop(), and next_bucket_loop().

6.4.2.6 unsigned int bucket_loop::head_hash

Definition at line 75 of file refobj.c.

Referenced by init_bucket_loop(), and next_bucket_loop().

6.4.2.7 int bucket_loop::version

Definition at line 74 of file refobj.c.

Referenced by init_bucket_loop(), next_bucket_loop(), and remove_bucket_loop().

The documentation for this struct was generated from the following file:

- src/refobj.c

6.5 config_category Struct Reference

Data Fields

- const char * name
- struct bucket_list * entries

6.5.1 Detailed Description

Definition at line 33 of file config.c.

6.5.2 Field Documentation

6.5.2.1 struct bucket_list* config_category::entries

Definition at line 35 of file config.c.

Referenced by get_category_next(), and get_config_category().

6.5.2.2 const char* config_category::name

Definition at line 34 of file config.c.

Referenced by get_category_next().

The documentation for this struct was generated from the following file:

- src/config.c

6.6 config_entry Struct Reference

```
#include <dtsapp.h>
```

Data Fields

- const char * item
- const char * value

6.6.1 Detailed Description

Definition at line 103 of file dtsapp.h.

6.6.2 Field Documentation

6.6.2.1 const char* config_entry::item

Definition at line 104 of file dtsapp.h.

6.6.2.2 const char* config_entry::value

Definition at line 105 of file dtsapp.h.

The documentation for this struct was generated from the following file:

- src/include/dtsapp.h

6.7 config_file Struct Reference

Data Fields

- const char * filename
- const char * filepath
- struct bucket_list * cat

6.7.1 Detailed Description

Definition at line 38 of file config.c.

6.7.2 Field Documentation

6.7.2.1 struct bucket_list* config_file::cat

Definition at line 41 of file config.c.

Referenced by get_config_file(), and process_config().

6.7.2.2 const char* config_file::filename

Definition at line 39 of file config.c.

6.7.2.3 const char* config_file::filepath

Definition at line 40 of file config.c.

Referenced by process_config().

The documentation for this struct was generated from the following file:

- src/config.c

6.8 curl_post Struct Reference

Data Fields

- struct curl_httppost * first
- struct curl_httppost * last

6.8.1 Detailed Description

Definition at line 33 of file curl.c.

6.8.2 Field Documentation

6.8.2.1 struct curl_httppost* curl_post::first

Definition at line 34 of file curl.c.

Referenced by curl_newpost(), curl_postitem(), and free_post().

6.8.2.2 struct curl_httppost* curl_post::last

Definition at line 35 of file curl.c.

Referenced by curl_newpost(), and curl_postitem().

The documentation for this struct was generated from the following file:

- src/curl.c

6.9 curlbuf Struct Reference

`#include <dtsapp.h>`

Data Fields

- uint8_t * header
- uint8_t * body
- char * c_type
- size_t hsize
- size_t bsize

6.9.1 Detailed Description

Definition at line 593 of file dtsapp.h.

6.9.2 Field Documentation

6.9.2.1 uint8_t* curlbuf::body

Definition at line 595 of file dtsapp.h.

Referenced by curl_buf2xml(), and curl_ungzip().

6.9.2.2 size_t curlbuf::bsize

Definition at line 598 of file dtsapp.h.

Referenced by curl_buf2xml(), and curl_ungzip().

6.9.2.3 char* curlbuf::c_type

Definition at line 596 of file dtsapp.h.

Referenced by curl_buf2xml().

6.9.2.4 uint8_t* curlbuf::header

Definition at line 594 of file dtsapp.h.

6.9.2.5 size_t curlbuf::hsize

Definition at line 597 of file dtsapp.h.

The documentation for this struct was generated from the following file:

- src/include/dtsapp.h

6.10 dn_naddr Struct Reference

`#include <utils.h>`

Data Fields

- unsigned short a_len
- unsigned char a_addr [DN_MAXADDL]

6.10.1 Detailed Description

Definition at line 59 of file utils.h.

6.10.2 Field Documentation

6.10.2.1 unsigned char dn_naddr::a_addr[DN_MAXADDL]

Definition at line 61 of file utils.h.

Referenced by get_addr_1(), and rt_addr_n2a().

6.10.2.2 unsigned short dn_naddr::a_len

Definition at line 60 of file utils.h.

The documentation for this struct was generated from the following file:

- src/libnetlink/include/utils.h

6.11 framework_core Struct Reference

Application framework data.

`#include <dtsapp.h>`

Data Fields

- const char * developer

  Developer/Copyright holder.
- const char * email

  Email address of copyright holder.
- const char * www

  URL displayed (use full URL ie with `http://`)

- const char * runfile
  File to write PID too and lock.
- const char * progname
  Detailed application name.
- int year
  Copyright year.
- int flock
  if there is a file locked this is the FD that will be unlocked and unlinked
- struct sigaction * sa
  sigaction structure allocated on execution
- syssighandler sig_handler
  Signal handler to pass signals too.
- int flags
  Application Options.

6.11.1 Detailed Description

Application framework data.

See Also

framework_mkcore()
framework_init()
FRAMEWORK_MAIN()

Definition at line 194 of file dtsapp.h.

6.11.2 Field Documentation

6.11.2.1 const char* framework_core::developer

Developer/Copyright holder.

Definition at line 196 of file dtsapp.h.

Referenced by framework_mkcore(), and printgnu().

6.11.2.2 const char* framework_core::email

Email address of copyright holder.

Definition at line 198 of file dtsapp.h.

Referenced by framework_mkcore(), and printgnu().

6.11.2.3 int framework_core::flags

Application Options.

See Also

application_flags

Definition at line 216 of file dtsapp.h.

Referenced by framework_init(), and framework_mkcore().

6.11.2.4 int framework_core::flock

if there is a file locked this is the FD that will be unlocked and unlinked

Definition at line 208 of file dtsapp.h.

Referenced by framework_init(), and lockpidfile().

6.11.2.5 const char* framework_core::progname

Detailed application name.

Definition at line 204 of file dtsapp.h.

Referenced by framework_mkcore(), and printgnu().

6.11.2.6 const char* framework_core::runfile

File to write PID too and lock.

Definition at line 202 of file dtsapp.h.

Referenced by framework_init(), and framework_mkcore().

6.11.2.7 struct sigaction* framework_core::sa

sigaction structure allocated on execution

Definition at line 210 of file dtsapp.h.

Referenced by framework_init(), and framework_mkcore().

6.11.2.8 syssighandler framework_core::sig_handler

Signal handler to pass signals too.

Note

The application framework installs a signal handler but will pass calls to this as a callback

Definition at line 213 of file dtsapp.h.

Referenced by framework_mkcore().

6.11.2.9 const char* framework_core::www

URL displayed (use full URL ie with `http://`)

Definition at line 200 of file dtsapp.h.

Referenced by framework_mkcore(), and printgnu().

6.11.2.10 int framework_core::year

Copyright year.

Definition at line 206 of file dtsapp.h.

Referenced by framework_mkcore(), and printgnu().

The documentation for this struct was generated from the following file:

- src/include/dtsapp.h

6.12 framework_sockthread Struct Reference

Unix socket data structure.

Data Fields

- char sock [UNIX_PATH_MAX+1]

  Socket path.
- int mask

  Socket umask.
- int protocol

  Socket protocol.
- threadfunc client

  Thread to begin on client connect.
- threadcleanup cleanup

  Thread clean up function.

6.12.1 Detailed Description

Unix socket data structure.

Definition at line 50 of file unixsock.c.

6.12.2 Field Documentation

6.12.2.1 threadcleanup framework_sockthread::cleanup

Thread clean up function.

See Also

threadcleanup

Definition at line 62 of file unixsock.c.

Referenced by framework_unixsocket().

6.12.2.2 threadfunc framework_sockthread::client

Thread to begin on client connect.

See Also

threadfunc

Definition at line 59 of file unixsock.c.

Referenced by framework_unixsocket().

6.12.2.3 int framework_sockthread::mask

Socket umask.

Definition at line 54 of file unixsock.c.

Referenced by framework_unixsocket().

6.12.2.4 int framework_sockthread::protocol

Socket protocol.

Definition at line 56 of file unixsock.c.

Referenced by framework_unixsocket().

6.12.2.5 char framework_sockthread::sock[UNIX_PATH_MAX+1]

Socket path.

Definition at line 52 of file unixsock.c.

Referenced by framework_unixsocket().

The documentation for this struct was generated from the following file:

- src/unixsock.c

6.13 fwsocket Struct Reference

`#include <dtsapp.h>`

Data Fields

- int sock
- int proto
- int type
- enum sock_flags flags
- union sockstruct addr
- struct ssldata * ssl
- struct fwsocket * parent
- struct bucket_list * children

6.13.1 Detailed Description

Definition at line 92 of file dtsapp.h.

6.13.2 Field Documentation

6.13.2.1 union sockstruct fwsocket::addr

Definition at line 97 of file dtsapp.h.

Referenced by dtls_listenssl().

6.13.2.2 struct bucket_list* fwsocket::children

Definition at line 100 of file dtsapp.h.

Referenced by socketserver().

6.13.2.3 enum sock_flags fwsocket::flags

Definition at line 96 of file dtsapp.h.

Referenced by socketread_d(), socketserver(), and socketwrite_d().

6.13.2.4 struct fwsocket* fwsocket::parent

Definition at line 99 of file dtsapp.h.

6.13.2.5 int fwsocket::proto

Definition at line 94 of file dtsapp.h.

Referenced by dtls_listenssl(), and make_socket().

6.13.2.6 int fwsocket::sock

Definition at line 93 of file dtsapp.h.

Referenced by dtls_listenssl(), make_socket(), radconnect(), socketread_d(), and socketwrite_d().

6.13.2.7 struct ssldata* fwsocket::ssl

Definition at line 98 of file dtsapp.h.

Referenced by dtls_listenssl(), dtlshandltimeout(), dtlstimeout(), dtsl_serveropts(), make_socket(), socketread_d(), socketserver(), socketwrite_d(), startsslclient(), and tlsaccept().

6.13.2.8 int fwsocket::type

Definition at line 95 of file dtsapp.h.

Referenced by dtls_listenssl(), make_socket(), socketread_d(), socketserver(), socketwrite_d(), and startsslclient().

The documentation for this struct was generated from the following file:

- src/include/dtsapp.h

6.14 hashedlist Struct Reference

`#include <list.h>`

Data Fields

- void * data
- int hash
- struct hashedlist * next
- struct hashedlist * prev

6.14.1 Detailed Description

Definition at line 34 of file list.h.

6.14.2 Field Documentation

6.14.2.1 void* hashedlist::data

Definition at line 35 of file list.h.

6.14.2.2 int hashedlist::hash

Definition at line 36 of file list.h.

6.14.2.3 struct hashedlist* hashedlist::next

Definition at line 37 of file list.h.

6.14.2.4 struct hashedlist* hashedlist::prev

Definition at line 38 of file list.h.

The documentation for this struct was generated from the following file:

- src/include/list.h

6.15 inet_prefix Struct Reference

`#include <utils.h>`

Data Fields

- __u8 family
- __u8 bytelen
- __s16 bitlen
- __u32 flags
- __u32 data [8]

6.15.1 Detailed Description

Definition at line 44 of file utils.h.

6.15.2 Field Documentation

6.15.2.1 __s16 inet_prefix::bitlen

Definition at line 47 of file utils.h.

Referenced by get_addr_1(), get_prefix_1(), and set_interface_ipaddr().

6.15.2.2 __u8 inet_prefix::bytelen

Definition at line 46 of file utils.h.

Referenced by get_addr_1(), get_prefix_1(), and set_interface_ipaddr().

6.15.2.3 __u32 inet_prefix::data[8]

Definition at line 49 of file utils.h.

Referenced by get_addr32(), get_addr_1(), inet_addr_match(), ll_addr_a2n(), and set_interface_ipaddr().

6.15.2.4 __u8 inet_prefix::family

Definition at line 45 of file utils.h.

Referenced by get_addr_1(), get_prefix_1(), and set_interface_ipaddr().

6.15.2.5 __u32 inet_prefix::flags

Definition at line 48 of file utils.h.

Referenced by get_prefix_1().

The documentation for this struct was generated from the following file:

- src/libnetlink/include/utils.h

6.16 ipaddr_req Struct Reference

Data Fields

- struct nlmsghdr n
- struct ifaddrmsg i
- char buf [1024]

6.16.1 Detailed Description

Definition at line 55 of file interface.c.

6.16.2 Field Documentation

6.16.2.1 char ipaddr_req::buf[1024]

Definition at line 58 of file interface.c.

6.16.2.2 struct ifaddrmsg ipaddr_req::i

Definition at line 57 of file interface.c.

Referenced by set_interface_ipaddr().

6.16.2.3 struct nlmsghdr ipaddr_req::n

Definition at line 56 of file interface.c.

Referenced by set_interface_ipaddr().

The documentation for this struct was generated from the following file:

- src/interface.c

6.17 iplink_req Struct Reference

Data Fields

- struct nlmsghdr n
- struct ifinfomsg i
- char buf [1024]

6.17.1 Detailed Description

Definition at line 49 of file interface.c.

6.17.2 Field Documentation

6.17.2.1 char iplink_req::buf[1024]

Definition at line 52 of file interface.c.

6.17.2.2 struct ifinfomsg iplink_req::i

Definition at line 51 of file interface.c.

Referenced by set_interface_addr(), set_interface_flags(), and set_interface_name().

6.17.2.3 struct nlmsghdr iplink_req::n

Definition at line 50 of file interface.c.

Referenced by create_kernmac(), create_kernvlan(), set_interface_addr(), set_interface_flags(), and set_interface_name().

The documentation for this struct was generated from the following file:

- src/interface.c

6.18 ipx_addr Struct Reference

`#include <utils.h>`

Data Fields

- u_int32_t ipx_net
- u_int8_t ipx_node [IPX_NODE_LEN]

6.18.1 Detailed Description

Definition at line 66 of file utils.h.

6.18.2 Field Documentation

6.18.2.1 u_int32_t ipx_addr::ipx_net

Definition at line 67 of file utils.h.

6.18.2.2 u_int8_t ipx_addr::ipx_node[IPX_NODE_LEN]

Definition at line 68 of file utils.h.

The documentation for this struct was generated from the following file:

- src/libnetlink/include/utils.h

6.19 ldap_add Struct Reference

Data Fields

- const char * dn
- struct bucket_list * bl

6.19.1 Detailed Description

Definition at line 46 of file openldap.c.

6.19.2 Field Documentation

6.19.2.1 struct bucket_list* ldap_add::bl

Definition at line 48 of file openldap.c.

Referenced by free_add(), getaddreq(), ldap_addinit(), and ldap_doadd().

6.19.2.2 const char* ldap_add::dn

Definition at line 47 of file openldap.c.

Referenced by free_add(), ldap_addinit(), and ldap_doadd().

The documentation for this struct was generated from the following file:

- src/openldap.c

6.20 ldap_attr Struct Reference

```
#include <dtsapp.h>
```

Data Fields

- const char * name
- int count
- struct ldap_attrval ** vals
- struct ldap_attr * next
- struct ldap_attr * prev

6.20.1 Detailed Description

Definition at line 522 of file dtsapp.h.

6.20.2 Field Documentation

6.20.2.1 int ldap_attr::count

Definition at line 524 of file dtsapp.h.

Referenced by attr2bl().

6.20.2.2 const char* ldap_attr::name

Definition at line 523 of file dtsapp.h.

Referenced by attr2bl(), free_attr(), and ldapattr_hash().

6.20.2.3 struct ldap_attr* ldap_attr::next

Definition at line 526 of file dtsapp.h.

Referenced by attr2bl(), free_attr(), free_entry(), and ldap_unref_attr().

6.20.2.4 struct ldap_attr* ldap_attr::prev

Definition at line 527 of file dtsapp.h.

Referenced by attr2bl(), and free_attr().

6.20.2.5 struct ldap_attrval** ldap_attr::vals

Definition at line 525 of file dtsapp.h.

Referenced by attr2bl(), and free_attr().

The documentation for this struct was generated from the following file:

- src/include/dtsapp.h

6.21 ldap_attrval Struct Reference

`#include <dtsapp.h>`

Data Fields

- int len
- enum ldap_attrtype type
- char * buffer

6.21.1 Detailed Description

Definition at line 516 of file dtsapp.h.

6.21.2 Field Documentation

6.21.2.1 char* ldap_attrval::buffer

Definition at line 519 of file dtsapp.h.

Referenced by attr2bl(), and free_attrval().

6.21.2.2 int ldap_attrval::len

Definition at line 517 of file dtsapp.h.

Referenced by attr2bl().

6.21.2.3 enum ldap_attrtype ldap_attrval::type

Definition at line 518 of file dtsapp.h.

Referenced by attr2bl().

The documentation for this struct was generated from the following file:

- src/include/dtsapp.h

6.22 ldap_conn Struct Reference

Data Fields

- LDAP * ldap
- char * uri
- int timelim
- int limit
- LDAPControl ** sctrlsp
- struct sasl_defaults * sasl
- struct ldap_simple * simple

6.22.1 Detailed Description

Definition at line 31 of file openldap.c.

6.22.2 Field Documentation

6.22.2.1 LDAP* ldap_conn::ldap

Definition at line 32 of file openldap.c.

Referenced by dts_ldapsearch(), free_ldapconn(), ldap_connect(), ldap_doadd(), ldap_domodify(), ldap_rebind_proc(), ldap_saslbind(), and ldap_simplebind().

6.22.2.2 int ldap_conn::limit

Definition at line 35 of file openldap.c.

Referenced by dts_ldapsearch(), and ldap_connect().

6.22.2.3 struct sasl_defaults* ldap_conn::sasl

Definition at line 37 of file openldap.c.

Referenced by free_ldapconn(), ldap_connect(), ldap_rebind_proc(), and ldap_saslbind().

6.22.2.4 LDAPControl** ldap_conn::sctrlsp

Definition at line 36 of file openldap.c.

Referenced by dts_ldapsearch(), free_ldapconn(), ldap_connect(), ldap_doadd(), ldap_domodify(), ldap_rebind_proc(), ldap_saslbind(), and ldap_simplebind().

6.22.2.5 struct ldap_simple* ldap_conn::simple

Definition at line 38 of file openldap.c.

Referenced by free_ldapconn(), ldap_rebind_proc(), and ldap_simplebind().

6.22.2.6 int ldap_conn::timelim

Definition at line 34 of file openldap.c.

Referenced by dts_ldapsearch(), and ldap_connect().

6.22.2.7 char* ldap_conn::uri

Definition at line 33 of file openldap.c.

Referenced by free_ldapconn(), and ldap_connect().

The documentation for this struct was generated from the following file:

- src/openldap.c

6.23 ldap_entry Struct Reference

```
#include <dtsapp.h>
```

Data Fields

- const char * dn
- const char * dnufn
- int rdncnt
- struct ldap_rdn ** rdn
- struct ldap_attr * list
- struct bucket_list * attrs
- struct ldap_attr * first_attr
- struct ldap_entry * next
- struct ldap_entry * prev

6.23.1 Detailed Description

Definition at line 530 of file dtsapp.h.

6.23.2 Field Documentation

6.23.2.1 struct bucket_list* ldap_entry::attrs

Definition at line 536 of file dtsapp.h.

Referenced by free_entry(), ldap_getattr(), ldap_getent(), and ldap_unref_attr().

6.23.2.2 const char* ldap_entry::dn

Definition at line 531 of file dtsapp.h.

Referenced by free_entry(), ldap_getent(), ldap_simplerebind(), and searchresults_hash().

6.23.2.3 const char* ldap_entry::dnufn

Definition at line 532 of file dtsapp.h.

Referenced by free_entry(), and ldap_getent().

6.23.2.4 struct ldap_attr* ldap_entry::first_attr

Definition at line 537 of file dtsapp.h.

Referenced by free_entry(), ldap_getent(), and ldap_unref_attr().

6.23.2.5 struct ldap_attr* ldap_entry::list

Definition at line 535 of file dtsapp.h.

6.23.2.6 struct ldap_entry* ldap_entry::next

Definition at line 538 of file dtsapp.h.

Referenced by dts_ldapsearch(), free_entry(), and ldap_unref_entry().

6.23.2.7 struct ldap_entry* ldap_entry::prev

Definition at line 539 of file dtsapp.h.

Referenced by dts_ldapsearch(), and free_entry().

6.23.2.8 struct ldap_rdn** ldap_entry::rdn

Definition at line 534 of file dtsapp.h.

Referenced by free_entry(), and ldap_getent().

6.23.2.9 int ldap_entry::rdncnt

Definition at line 533 of file dtsapp.h.

Referenced by ldap_getent().

The documentation for this struct was generated from the following file:

- src/include/dtsapp.h

6.24 ldap_modify Struct Reference

Data Fields

- const char * dn
- struct bucket_list * bl [3]

6.24.1 Detailed Description

Definition at line 41 of file openldap.c.

6.24.2 Field Documentation

6.24.2.1 struct bucket_list* ldap_modify::bl[3]

Definition at line 43 of file openldap.c.

Referenced by free_modify(), getmodreq(), ldap_domodify(), and ldap_modifyinit().

6.24.2.2 const char* ldap_modify::dn

Definition at line 42 of file openldap.c.

Referenced by free_modify(), ldap_domodify(), and ldap_modifyinit().

The documentation for this struct was generated from the following file:

- src/openldap.c

6.25 ldap_modreq Struct Reference

Data Fields

- const char * attr
- int cnt
- struct ldap_modval * first
- struct ldap_modval * last

6.25.1 Detailed Description

Definition at line 56 of file openldap.c.

6.25.2 Field Documentation

6.25.2.1 const char* ldap_modreq::attr

Definition at line 57 of file openldap.c.

Referenced by free_modreq(), ldap_reqtoarr(), modify_hash(), and new_modreq().

6.25.2.2 int ldap_modreq::cnt

Definition at line 58 of file openldap.c.

Referenced by add_modifyval(), ldap_domodify(), and ldap_reqtoarr().

6.25.2.3 struct ldap_modval* ldap_modreq::first

Definition at line 59 of file openldap.c.

Referenced by add_modifyval(), free_modreq(), and ldap_reqtoarr().

6.25.2.4 struct ldap_modval* ldap_modreq::last

Definition at line 60 of file openldap.c.

Referenced by add_modifyval().

The documentation for this struct was generated from the following file:

- src/openldap.c

6.26 ldap_modval Struct Reference

Data Fields

- const char * value
- struct ldap_modval * next

6.26.1 Detailed Description

Definition at line 51 of file openldap.c.

6.26.2 Field Documentation

6.26.2.1 struct ldap_modval* ldap_modval::next

Definition at line 53 of file openldap.c.

Referenced by add_modifyval(), free_modreq(), and ldap_reqtoarr().

6.26.2.2 const char* ldap_modval::value

Definition at line 52 of file openldap.c.

Referenced by add_modifyval(), free_modval(), and ldap_reqtoarr().

The documentation for this struct was generated from the following file:

- src/openldap.c

6.27 ldap_rdn Struct Reference

`#include <dtsapp.h>`

Data Fields

- const char * name
- const char * value
- struct ldap_rdn * next
- struct ldap_rdn * prev

6.27.1 Detailed Description

Definition at line 509 of file dtsapp.h.

6.27.2 Field Documentation

6.27.2.1 const char* ldap_rdn::name

Definition at line 510 of file dtsapp.h.

Referenced by free_rdn(), and ldap_getent().

6.27.2.2 struct ldap_rdn* ldap_rdn::next

Definition at line 512 of file dtsapp.h.

Referenced by ldap_getent().

6.27.2.3 struct ldap_rdn* ldap_rdn::prev

Definition at line 513 of file dtsapp.h.

Referenced by ldap_getent().

6.27.2.4 const char* ldap_rdn::value

Definition at line 511 of file dtsapp.h.

Referenced by free_rdn(), and ldap_getent().

The documentation for this struct was generated from the following file:

- src/include/dtsapp.h

6.28 ldap_results Struct Reference

```
#include <dtsapp.h>
```

Data Fields

- int count
- struct ldap_entry * first_entry
- struct bucket_list * entries

6.28.1 Detailed Description

Definition at line 542 of file dtsapp.h.

6.28.2 Field Documentation

6.28.2.1 int ldap_results::count

Definition at line 543 of file dtsapp.h.

Referenced by dts_ldapsearch(), and ldap_simplerebind().

6.28.2.2 struct bucket_list* ldap_results::entries

Definition at line 545 of file dtsapp.h.

Referenced by dts_ldapsearch(), free_result(), ldap_getentry(), and ldap_unref_entry().

6.28.2.3 struct ldap_entry* ldap_results::first_entry

Definition at line 544 of file dtsapp.h.

Referenced by dts_ldapsearch(), ldap_simplerebind(), and ldap_unref_entry().

The documentation for this struct was generated from the following file:

- src/include/dtsapp.h

6.29 ldap_simple Struct Reference

Data Fields

- const char * dn
- struct berval * cred

6.29.1 Detailed Description

Definition at line 26 of file openldap.c.

6.29.2 Field Documentation

6.29.2.1 struct berval* ldap_simple::cred

Definition at line 28 of file openldap.c.

Referenced by free_simple(), ldap_rebind_proc(), and ldap_simplebind().

6.29.2.2 const char* ldap_simple::dn

Definition at line 27 of file openldap.c.

Referenced by free_simple(), ldap_rebind_proc(), and ldap_simplebind().

The documentation for this struct was generated from the following file:

- src/openldap.c

6.30 linkedlist Struct Reference

`#include <list.h>`

Data Fields

- void * data
- struct linkedlist * next
- struct linkedlist * prev

6.30.1 Detailed Description

Definition at line 28 of file list.h.

6.30.2 Field Documentation

6.30.2.1 void* linkedlist::data

Definition at line 29 of file list.h.

6.30.2.2 struct linkedlist* linkedlist::next

Definition at line 30 of file list.h.

6.30.2.3 struct linkedlist* linkedlist::prev

Definition at line 31 of file list.h.

The documentation for this struct was generated from the following file:

- src/include/list.h

6.31 ll_cache Struct Reference

Data Fields

- struct ll_cache * idx_next
- unsigned flags
- int index
- unsigned short type
- unsigned short alen
- char name [IFNAMSIZ]
- unsigned char addr [20]

6.31.1 Detailed Description

Definition at line 28 of file ll_map.c.

6.31.2 Field Documentation

6.31.2.1 unsigned char ll_cache::addr[20]

Definition at line 35 of file ll_map.c.

Referenced by ll_index_to_addr(), and ll_remember_index().

6.31.2.2 unsigned short ll_cache::alen

Definition at line 33 of file ll_map.c.

Referenced by ll_index_to_addr(), and ll_remember_index().

6.31.2.3 unsigned ll_cache::flags

Definition at line 30 of file ll_map.c.

Referenced by ll_index_to_flags(), and ll_remember_index().

6.31.2.4 struct ll_cache* ll_cache::idx_next

Definition at line 29 of file ll_map.c.

Referenced by ll_idx_n2a(), ll_index_to_addr(), ll_index_to_flags(), ll_index_to_type(), ll_name_to_index(), and ll_remember_index().

6.31.2.5 int ll_cache::index

Definition at line 31 of file ll_map.c.

Referenced by ll_idx_n2a(), ll_index_to_addr(), ll_index_to_flags(), ll_index_to_type(), ll_name_to_index(), and ll_remember_index().

6.31.2.6 char ll_cache::name[IFNAMSIZ]

Definition at line 34 of file ll_map.c.

Referenced by ll_idx_n2a(), ll_name_to_index(), and ll_remember_index().

6.31.2.7 unsigned short ll_cache::type

Definition at line 32 of file ll_map.c.

Referenced by ll_index_to_type(), and ll_remember_index().

The documentation for this struct was generated from the following file:

- src/libnetlink/ll_map.c

6.32 natmap Struct Reference

Data Fields

- uint16_t mask
- uint16_t adjo
- uint16_t adji
- uint8_t ipre [16]
- uint8_t epre [16]
- uint32_t hash

6.32.1 Detailed Description

Definition at line 26 of file rfc6296.c.

6.32.2 Field Documentation

6.32.2.1 uint16_t natmap::adji

Definition at line 29 of file rfc6296.c.

Referenced by rfc6296_map(), and rfc6296_map_add().

6.32.2.2 uint16_t natmap::adjo

Definition at line 28 of file rfc6296.c.

Referenced by rfc6296_map(), and rfc6296_map_add().

6.32.2.3 uint8_t natmap::epre[16]

Definition at line 31 of file rfc6296.c.

Referenced by rfc6296_map(), and rfc6296_map_add().

6.32.2.4 uint32_t natmap::hash

Definition at line 32 of file rfc6296.c.

6.32.2.5 uint8_t natmap::ipre[16]

Definition at line 30 of file rfc6296.c.

Referenced by rfc6296_map(), and rfc6296_map_add().

6.32.2.6 uint16_t natmap::mask

Definition at line 27 of file rfc6296.c.

Referenced by rfc6296_map(), and rfc6296_map_add().

The documentation for this struct was generated from the following file:

- src/rfc6296.c

6.33 nfct_struct Struct Reference

Data Fields

- struct nfct_handle * nfct
- int fd
- int flags

6.33.1 Detailed Description

Definition at line 41 of file nf_ctrack.c.

6.33.2 Field Documentation

6.33.2.1 int nfct_struct::fd

Definition at line 43 of file nf_ctrack.c.

6.33.2.2 int nfct_struct::flags

Definition at line 44 of file nf_ctrack.c.

6.33.2.3 struct nfct_handle* nfct_struct::nfct

Definition at line 42 of file nf_ctrack.c.

Referenced by nf_ctrack_delete(), nf_ctrack_dump(), nf_ctrack_nat(), and nf_ctrack_trace().

The documentation for this struct was generated from the following file:

- src/nf_ctrack.c

6.34 nfq_list Struct Reference

Data Fields

- struct bucket_list * queues

6.34.1 Detailed Description

Definition at line 56 of file nf_queue.c.

6.34.2 Field Documentation

6.34.2.1 struct bucket_list* nfq_list::queues

Definition at line 57 of file nf_queue.c.

Referenced by nfqueue_attach().

The documentation for this struct was generated from the following file:

- src/nf_queue.c

6.35 nfq_queue Struct Reference

Data Fields

- struct nfq_struct * nfq
- struct nfq_q_handle * qh
- nfqueue_cb cb
- void * data
- uint16_t num

6.35.1 Detailed Description

Definition at line 48 of file nf_queue.c.

6.35.2 Field Documentation

6.35.2.1 nfqueue_cb nfq_queue::cb

Definition at line 51 of file nf_queue.c.

Referenced by nfqueue_attach().

6.35.2.2 void* nfq_queue::data

Definition at line 52 of file nf_queue.c.

Referenced by nfqueue_attach().

6.35.2.3 struct nfq_struct* nfq_queue::nfq

Definition at line 49 of file nf_queue.c.

Referenced by nfqueue_attach().

6.35.2.4 uint16_t nfq_queue::num

Definition at line 53 of file nf_queue.c.

6.35.2.5 struct nfq_q_handle* nfq_queue::qh

Definition at line 50 of file nf_queue.c.

Referenced by nfqueue_attach().

The documentation for this struct was generated from the following file:

- src/nf_queue.c

6.36 nfq_struct Struct Reference

Data Fields

- struct nfq_handle * h
- uint16_t pf
- int fd
- int flags

6.36.1 Detailed Description

Definition at line 41 of file nf_queue.c.

6.36.2 Field Documentation

6.36.2.1 int nfq_struct::fd

Definition at line 44 of file nf_queue.c.

6.36.2.2 int nfq_struct::flags

Definition at line 45 of file nf_queue.c.

6.36.2.3 struct nfq_handle* nfq_struct::h

Definition at line 42 of file nf_queue.c.

Referenced by nfqueue_attach().

6.36.2.4 uint16_t nfq_struct::pf

Definition at line 43 of file nf_queue.c.

The documentation for this struct was generated from the following file:

- src/nf_queue.c

6.37 pseudohdr Struct Reference

Data Fields

- uint32_t saddr

- uint32_t daddr
- uint8_t zero
- uint8_t proto
- uint16_t len

6.37.1 Detailed Description

Definition at line 74 of file iputil.c.

6.37.2 Field Documentation

6.37.2.1 uint32_t pseudohdr::daddr

Definition at line 76 of file iputil.c.

Referenced by ipv4tcpchecksum(), and ipv4udpchecksum().

6.37.2.2 uint16_t pseudohdr::len

Definition at line 79 of file iputil.c.

Referenced by ipv4tcpchecksum(), and ipv4udpchecksum().

6.37.2.3 uint8_t pseudohdr::proto

Definition at line 78 of file iputil.c.

Referenced by ipv4tcpchecksum(), and ipv4udpchecksum().

6.37.2.4 uint32_t pseudohdr::saddr

Definition at line 75 of file iputil.c.

Referenced by ipv4tcpchecksum(), and ipv4udpchecksum().

6.37.2.5 uint8_t pseudohdr::zero

Definition at line 77 of file iputil.c.

Referenced by ipv4tcpchecksum(), and ipv4udpchecksum().

The documentation for this struct was generated from the following file:

- src/iputil.c

6.38 radius_connection Struct Reference

Data Fields

- struct fwsocket * socket
- unsigned char id
- struct radius_server * server
- int flags
- struct bucket_list * sessions

6.38.1 Detailed Description

Definition at line 67 of file radius.c.

6.38.2 Field Documentation

6.38.2.1 int radius_connection::flags

Definition at line 71 of file radius.c.

6.38.2.2 unsigned char radius_connection::id

Definition at line 69 of file radius.c.

Referenced by radconnect().

6.38.2.3 struct radius_server* radius_connection::server

Definition at line 70 of file radius.c.

Referenced by radconnect().

6.38.2.4 struct bucket_list* radius_connection::sessions

Definition at line 72 of file radius.c.

6.38.2.5 struct fwsocket* radius_connection::socket

Definition at line 68 of file radius.c.

Referenced by radconnect().

The documentation for this struct was generated from the following file:

- src/radius.c

6.39 radius_packet Struct Reference

Data Fields

- unsigned char code
- unsigned char id
- unsigned short len
- unsigned char token [RAD_AUTH_TOKEN_LEN]
- unsigned char attrs [RAD_AUTH_PACKET_LEN-RAD_AUTH_HDR_LEN]

6.39.1 Detailed Description

Definition at line 39 of file radius.c.

6.39.2 Field Documentation

6.39.2.1 unsigned char radius_packet::attrs[RAD_AUTH_PACKET_LEN-RAD_AUTH_HDR_LEN]

Definition at line 44 of file radius.c.

Referenced by addradattr(), radius_attr_first(), and radius_attr_next().

6.39.2.2 unsigned char radius_packet::code

Definition at line 40 of file radius.c.

Referenced by new_radpacket().

6.39.2.3 unsigned char radius_packet::id

Definition at line 41 of file radius.c.

6.39.2.4 unsigned short radius_packet::len

Definition at line 42 of file radius.c.

Referenced by addradattr(), new_radpacket(), and radius_attr_next().

6.39.2.5 unsigned char radius_packet::token[RAD_AUTH_TOKEN_LEN]

Definition at line 43 of file radius.c.

Referenced by new_radpacket().

The documentation for this struct was generated from the following file:

- src/radius.c

6.40 radius_server Struct Reference

Data Fields

- const char * name
- const char * authport
- const char * acctport
- const char * secret
- unsigned char id
- int timeout
- struct timeval service
- struct bucket_list * connex

6.40.1 Detailed Description

Definition at line 80 of file radius.c.

6.40.2 Field Documentation

6.40.2.1 const char* radius_server::acctport

Definition at line 83 of file radius.c.

Referenced by add_radserver().

6.40.2.2 const char* radius_server::authport

Definition at line 82 of file radius.c.

Referenced by add_radserver(), and radconnect().

6.40.2.3 struct bucket_list* radius_server::connex

Definition at line 88 of file radius.c.

Referenced by radconnect().

6.40.2.4 unsigned char radius_server::id

Definition at line 85 of file radius.c.

Referenced by add_radserver().

6.40.2.5 const char* radius_server::name

Definition at line 81 of file radius.c.

Referenced by add_radserver(), and radconnect().

6.40.2.6 const char* radius_server::secret

Definition at line 84 of file radius.c.

Referenced by add_radserver().

6.40.2.7 struct timeval radius_server::service

Definition at line 87 of file radius.c.

Referenced by add_radserver().

6.40.2.8 int radius_server::timeout

Definition at line 86 of file radius.c.

Referenced by add_radserver().

The documentation for this struct was generated from the following file:

- src/radius.c

6.41 radius_session Struct Reference

Data Fields

- unsigned short id
- unsigned char request [RAD_AUTH_TOKEN_LEN]
- void * cb_data
- radius_cb read_cb
- unsigned int olen
- struct radius_packet * packet
- struct timeval sent
- const char * passwd
- char retries
- char minserver

6.41.1 Detailed Description

Definition at line 51 of file radius.c.

6.41.2 Field Documentation

6.41.2.1 void* radius_session::cb_data

Definition at line 54 of file radius.c.

6.41.2.2 unsigned short radius_session::id

Definition at line 52 of file radius.c.

6.41.2.3 char radius_session::minserver

Definition at line 61 of file radius.c.

6.41.2.4 unsigned int radius_session::olen

Definition at line 56 of file radius.c.

6.41.2.5 struct radius_packet* radius_session::packet

Definition at line 57 of file radius.c.

6.41.2.6 const char* radius_session::passwd

Definition at line 59 of file radius.c.

6.41.2.7 radius_cb radius_session::read_cb

Definition at line 55 of file radius.c.

6.41.2.8 unsigned char radius_session::request[RAD_AUTH_TOKEN_LEN]

Definition at line 53 of file radius.c.

6.41.2.9 char radius_session::retries

Definition at line 60 of file radius.c.

6.41.2.10 struct timeval radius_session::sent

Definition at line 58 of file radius.c.

The documentation for this struct was generated from the following file:

- src/radius.c

6.42 ref_obj Struct Reference

Data Fields

- int magic
- int cnt
- int size
- pthread_mutex_t * lock
- objdestroy destroy
- void * data

6.42.1 Detailed Description

Definition at line 38 of file refobj.c.

6.42.2 Field Documentation

6.42.2.1 int ref_obj::cnt

Definition at line 40 of file refobj.c.

Referenced by objalloc(), objcnt(), objref(), and objunref().

6.42.2.2 void* ref_obj::data

Definition at line 44 of file refobj.c.

Referenced by addtobucket(), bucket_list_find_key(), next_bucket_loop(), objalloc(), objcnt(), objlock(), objref(), objsize(), objtrylock(), objunlock(), objunref(), remove_bucket_item(), and remove_bucket_loop().

6.42.2.3 objdestroy ref_obj::destroy

Definition at line 43 of file refobj.c.

Referenced by objalloc(), and objunref().

6.42.2.4 pthread_mutex_t* ref_obj::lock

Definition at line 42 of file refobj.c.

Referenced by objalloc(), objcnt(), objlock(), objref(), objsize(), objtrylock(), objunlock(), and objunref().

6.42.2.5 int ref_obj::magic

Definition at line 39 of file refobj.c.

Referenced by objalloc(), objcnt(), objlock(), objref(), objsize(), objtrylock(), objunlock(), and objunref().

6.42.2.6 int ref_obj::size

Definition at line 41 of file refobj.c.

Referenced by objalloc(), objsize(), and objunref().

The documentation for this struct was generated from the following file:

- src/refobj.c

6.43 rtnl_dump_filter_arg Struct Reference

`#include <libnetlink.h>`

Data Fields

- rtnl_filter_t filter
- void * arg1

6.43.1 Detailed Description

Definition at line 31 of file libnetlink.h.

6.43.2 Field Documentation

6.43.2.1 void * rtnl_dump_filter_arg::arg1

Definition at line 33 of file libnetlink.h.

Referenced by rtnl_dump_filter(), and rtnl_dump_filter_l().

6.43.2.2 rtnl_filter_t rtnl_dump_filter_arg::filter

Definition at line 32 of file libnetlink.h.

Referenced by rtnl_dump_filter(), and rtnl_dump_filter_l().

The documentation for this struct was generated from the following files:

- src/libnetlink/include/libnetlink.h
- src/libnetlink/libnetlink.h

6.44 rtnl_handle Struct Reference

`#include <libnetlink.h>`

Data Fields

- int fd
- struct sockaddr_nl local
- struct sockaddr_nl peer
- __u32 seq
- __u32 dump

6.44.1 Detailed Description

Definition at line 12 of file libnetlink.h.

6.44.2 Field Documentation

6.44.2.1 __u32 rtnl_handle::dump

Definition at line 17 of file libnetlink.h.

Referenced by rtnl_dump_filter_l(), rtnl_dump_request(), and rtnl_wilddump_request().

6.44.2.2 int rtnl_handle::fd

Definition at line 13 of file libnetlink.h.

Referenced by rtnl_close(), rtnl_dump_filter_l(), rtnl_dump_request(), rtnl_listen(), rtnl_open_byproto(), rtnl_send(), rtnl_send_check(), rtnl_talk(), and rtnl_wilddump_request().

6.44.2.3 struct sockaddr_nl rtnl_handle::local

Definition at line 14 of file libnetlink.h.

Referenced by rtnl_dump_filter_l(), rtnl_open_byproto(), and rtnl_talk().

6.44.2.4 struct sockaddr_nl rtnl_handle::peer

Definition at line 15 of file libnetlink.h.

6.44.2.5 __u32 rtnl_handle::seq

Definition at line 16 of file libnetlink.h.

Referenced by rtnl_dump_request(), rtnl_open_byproto(), rtnl_talk(), and rtnl_wilddump_request().

The documentation for this struct was generated from the following files:

- src/libnetlink/include/libnetlink.h
- src/libnetlink/libnetlink.h

6.45 rtnl_hash_entry Struct Reference

Data Fields

- struct rtnl_hash_entry * next
- char * name
- unsigned int id

6.45.1 Detailed Description

Definition at line 30 of file rt_names.c.

6.45.2 Field Documentation

6.45.2.1 unsigned int rtnl_hash_entry::id

Definition at line 33 of file rt_names.c.

Referenced by rtnl_group_a2n(), rtnl_rttable_a2n(), and rtnl_rttable_n2a().

6.45.2.2 char* rtnl_hash_entry::name

Definition at line 32 of file rt_names.c.

Referenced by rtnl_group_a2n(), rtnl_rttable_a2n(), and rtnl_rttable_n2a().

6.45.2.3 struct rtnl_hash_entry* rtnl_hash_entry::next

Definition at line 31 of file rt_names.c.

Referenced by rtnl_group_a2n(), rtnl_rttable_a2n(), and rtnl_rttable_n2a().

The documentation for this struct was generated from the following file:

- src/libnetlink/rt_names.c

6.46 sasl_defaults Struct Reference

Data Fields

- const char * mech
- const char * realm
- const char * authcid
- const char * passwd
- const char * authzid

6.46.1 Detailed Description

Definition at line 18 of file openldap.c.

6.46.2 Field Documentation

6.46.2.1 const char* sasl_defaults::authcid

Definition at line 21 of file openldap.c.

Referenced by free_sasl(), and ldap_saslbind().

6.46.2.2 const char* sasl_defaults::authzid

Definition at line 23 of file openldap.c.

Referenced by free_sasl(), and ldap_saslbind().

6.46.2.3 const char* sasl_defaults::mech

Definition at line 19 of file openldap.c.

Referenced by free_sasl(), ldap_rebind_proc(), and ldap_saslbind().

6.46.2.4 const char* sasl_defaults::passwd

Definition at line 22 of file openldap.c.

Referenced by free_sasl(), and ldap_saslbind().

6.46.2.5 const char* sasl_defaults::realm

Definition at line 20 of file openldap.c.

Referenced by free_sasl(), and ldap_saslbind().

The documentation for this struct was generated from the following file:

- src/openldap.c

6.47 socket_handler Struct Reference

Data Fields

- struct fwsocket * sock
- void * data
- socketrecv client
- threadcleanup cleanup
- socketrecv connect

6.47.1 Detailed Description

Definition at line 39 of file socket.c.

6.47.2 Field Documentation

6.47.2.1 threadcleanup socket_handler::cleanup

Definition at line 43 of file socket.c.

6.47.2.2 socketrecv socket_handler::client

Definition at line 42 of file socket.c.

6.47.2.3 socketrecv socket_handler::connect

Definition at line 44 of file socket.c.

6.47.2.4 void* socket_handler::data

Definition at line 41 of file socket.c.

6.47.2.5 struct fwsocket* socket_handler::sock

Definition at line 40 of file socket.c.

The documentation for this struct was generated from the following file:

- src/socket.c

6.48 sockstruct Union Reference

`#include <dtsapp.h>`

Data Fields

- struct sockaddr sa
- struct sockaddr_in sa4
- struct sockaddr_in6 sa6
- struct sockaddr_storage ss

6.48.1 Detailed Description

Definition at line 77 of file dtsapp.h.

6.48.2 Field Documentation

6.48.2.1 struct sockaddr sockstruct::sa

Definition at line 78 of file dtsapp.h.

Referenced by dtls_listenssl(), socketread_d(), and socketwrite_d().

6.48.2.2 struct sockaddr_in sockstruct::sa4

Definition at line 79 of file dtsapp.h.

6.48.2.3 struct sockaddr_in6 sockstruct::sa6

Definition at line 80 of file dtsapp.h.

6.48.2.4 struct sockaddr_storage sockstruct::ss

Definition at line 81 of file dtsapp.h.

The documentation for this union was generated from the following file:

- src/include/dtsapp.h

6.49 ssldata Struct Reference

Data Fields

- SSL_CTX * ctx
- SSL * ssl
- BIO * bio
- int flags
- const SSL_METHOD * meth
- struct ssldata * parent

6.49.1 Detailed Description

Definition at line 46 of file sslutil.c.

6.49.2 Field Documentation

6.49.2.1 BIO* ssldata::bio

Definition at line 49 of file sslutil.c.

6.49.2.2 SSL_CTX* ssldata::ctx

Definition at line 47 of file sslutil.c.

Referenced by dtlsv1_init(), and dtsl_serveropts().

6.49.2.3 int ssldata::flags

Definition at line 50 of file sslutil.c.

Referenced by dtls_listenssl(), dtsl_serveropts(), and startsslclient().

6.49.2.4 const SSL_METHOD* ssldata::meth

Definition at line 51 of file sslutil.c.

6.49.2.5 struct ssldata* ssldata::parent

Definition at line 52 of file sslutil.c.

6.49.2.6 SSL* ssldata::ssl

Definition at line 48 of file sslutil.c.

Referenced by dtls_listenssl(), dtlshandltimeout(), dtlstimeout(), dtlsv1_init(), dtsl_serveropts(), socketread_d(), socketwrite_d(), ssl_shutdown(), and sslv3_init().

The documentation for this struct was generated from the following file:

- src/sslutil.c

6.50 thread_pvt Struct Reference

thread struct used to create threads data needs to be first element

Data Fields

- void * data

  Reference to data held on thread creation.
- int magic

  Magic number.
- pthread_t thr

  Thread information.
- threadcleanup cleanup

  Thread cleanup callback.
- threadfunc func

  Thread function.
- threadsighandler sighandler

  Thread signal handler.
- enum threadopt flags

  thread options

6.50.1 Detailed Description

thread struct used to create threads data needs to be first element

Definition at line 59 of file thread.c.

6.50.2 Field Documentation

6.50.2.1 threadcleanup thread_pvt::cleanup

Thread cleanup callback.

See Also

threadcleanup

Definition at line 68 of file thread.c.

Referenced by framework_mkthread().

6.50.2.2 void* thread_pvt::data

Reference to data held on thread creation.

Definition at line 61 of file thread.c.

Referenced by framework_mkthread(), and framework_threadok().

6.50.2.3 enum threadopt thread_pvt::flags

thread options

See Also

threadopt_flags

Definition at line 77 of file thread.c.

Referenced by framework_mkthread().

6.50.2.4 threadfunc thread_pvt::func

Thread function.

See Also

threadfunc

Definition at line 71 of file thread.c.

Referenced by framework_mkthread().

6.50.2.5 int thread_pvt::magic

Magic number.

Definition at line 63 of file thread.c.

Referenced by framework_mkthread(), and framework_threadok().

6.50.2.6 threadsighandler thread_pvt::sighandler

Thread signal handler.

See Also

threadsighandler

Definition at line 74 of file thread.c.

Referenced by framework_mkthread().

6.50.2.7 pthread_t thread_pvt::thr

Thread information.

Definition at line 65 of file thread.c.

Referenced by framework_mkthread(), framework_threadok(), and jointhreads().

The documentation for this struct was generated from the following file:

- src/thread.c

6.51 threadcontainer Struct Reference

Global threads data.

Data Fields

- struct bucket_list * list

  Hashed bucket list of threads.
- struct thread_pvt * manager

  Manager thread.

6.51.1 Detailed Description

Global threads data.

Definition at line 81 of file thread.c.

6.51.2 Field Documentation

6.51.2.1 struct bucket_list* threadcontainer::list

Hashed bucket list of threads.

Definition at line 83 of file thread.c.

Referenced by framework_mkthread(), and startthreads().

6.51.2.2 struct thread_pvt* threadcontainer::manager

Manager thread.

Definition at line 85 of file thread.c.

Referenced by framework_mkthread(), jointhreads(), startthreads(), and stopthreads().

The documentation for this struct was generated from the following file:

- src/thread.c

6.52 xml_attr Struct Reference

`#include <dtsapp.h>`

Data Fields

- const char * name
- const char * value

6.52.1 Detailed Description

Definition at line 451 of file dtsapp.h.

6.52.2 Field Documentation

6.52.2.1 const char* xml_attr::name

Definition at line 452 of file dtsapp.h.

Referenced by attr_hash(), and xml_nodetohash().

6.52.2.2 const char* xml_attr::value

Definition at line 453 of file dtsapp.h.

Referenced by xml_getattr(), and xml_nodetohash().

The documentation for this struct was generated from the following file:

- src/include/dtsapp.h

6.53 xml_buffer Struct Reference

`#include <priv_xml.h>`

Data Fields

- xmlChar * buffer
- int size

6.53.1 Detailed Description

Definition at line 22 of file priv_xml.h.

6.53.2 Field Documentation

6.53.2.1 xmlChar* xml_buffer::buffer

Definition at line 23 of file priv_xml.h.

Referenced by free_buffer(), xml_doctobuffer(), xml_getbuffer(), and xslt_apply_buffer().

6.53.2.2 int xml_buffer::size

Definition at line 24 of file priv_xml.h.

Referenced by xml_doctobuffer(), xml_modify2(), and xslt_apply_buffer().

The documentation for this struct was generated from the following file:

- src/include/priv_xml.h

6.54 xml_doc Struct Reference

`#include <priv_xml.h>`

Data Fields

- xmlDocPtr doc
- xmlNodePtr root
- xmlXPathContextPtr xpathCtx
- xmlValidCtxtPtr ValidCtxt

6.54.1 Detailed Description

Definition at line 27 of file priv_xml.h.

6.54.2 Field Documentation

6.54.2.1 xmlDocPtr xml_doc::doc

Definition at line 28 of file priv_xml.h.

Referenced by xml_addnode(), xml_doctobuffer(), xml_loadbuf(), xml_loaddoc(), xml_modify(), xml_nodetohash(), xml_savefile(), xml_setattr(), xslt_apply(), and xslt_apply_buffer().

6.54.2.2 xmlNodePtr xml_doc::root

Definition at line 29 of file priv_xml.h.

Referenced by xml_createpath(), xml_getrootname(), and xml_getrootnode().

6.54.2.3 xmlValidCtxtPtr xml_doc::ValidCtxt

Definition at line 31 of file priv_xml.h.

6.54.2.4 xmlXPathContextPtr xml_doc::xpathCtx

Definition at line 30 of file priv_xml.h.

Referenced by xml_createpath(), and xml_xpath().

The documentation for this struct was generated from the following file:

- src/include/priv_xml.h

6.55 xml_node Struct Reference

`#include <dtsapp.h>`

Data Fields

- const char * name
- const char * value
- const char * key
- struct bucket_list * attrs
- void * nodeptr

6.55.1 Detailed Description

Definition at line 456 of file dtsapp.h.

6.55.2 Field Documentation

6.55.2.1 struct bucket_list* xml_node::attrs

Definition at line 460 of file dtsapp.h.

Referenced by xml_getattr(), and xml_nodetohash().

6.55.2.2 const char* xml_node::key

Definition at line 459 of file dtsapp.h.

Referenced by node_hash(), and xml_nodetohash().

6.55.2.3 const char* xml_node::name

Definition at line 457 of file dtsapp.h.

Referenced by xml_nodetohash().

6.55.2.4 void* xml_node::nodeptr

Definition at line 461 of file dtsapp.h.

Referenced by xml_appendnode(), xml_delete(), xml_modify(), xml_modify2(), xml_nodetohash(), xml_setattr(), and xml_unlink().

6.55.2.5 const char* xml_node::value

Definition at line 458 of file dtsapp.h.

Referenced by xml_modify(), and xml_nodetohash().

The documentation for this struct was generated from the following file:

- src/include/dtsapp.h

6.56 xml_node_iter Struct Reference

Data Fields

- struct xml_search * xsearch
- int curpos
- int cnt

6.56.1 Detailed Description

Definition at line 25 of file libxml2.c.

6.56.2 Field Documentation

6.56.2.1 int xml_node_iter::cnt

Definition at line 28 of file libxml2.c.

Referenced by xml_getfirstnode(), and xml_getnextnode().

6.56.2.2 int xml_node_iter::curpos

Definition at line 27 of file libxml2.c.

Referenced by xml_getfirstnode(), and xml_getnextnode().

6.56.2.3 struct xml_search* xml_node_iter::xsearch

Definition at line 26 of file libxml2.c.

Referenced by xml_getfirstnode(), and xml_getnextnode().

The documentation for this struct was generated from the following file:

- src/libxml2.c

6.57 xml_search Struct Reference

Data Fields

- struct xml_doc * xmldoc
- xmlXPathObjectPtr xpathObj
- struct bucket_list * nodes

6.57.1 Detailed Description

Definition at line 31 of file libxml2.c.

6.57.2 Field Documentation

6.57.2.1 struct bucket_list* xml_search::nodes

Definition at line 34 of file libxml2.c.

Referenced by xml_getnode(), xml_getnodes(), and xml_xpath().

6.57.2.2 struct xml_doc* xml_search::xmldoc

Definition at line 32 of file libxml2.c.

Referenced by xml_gethash(), and xml_xpath().

6.57.2.3 xmlXPathObjectPtr xml_search::xpathObj

Definition at line 33 of file libxml2.c.

Referenced by xml_gethash(), xml_modify2(), xml_nodecount(), and xml_xpath().

The documentation for this struct was generated from the following file:

- src/libxml2.c

6.58 xslt_doc Struct Reference

Data Fields

- xsltStylesheetPtr doc
- struct bucket_list * params

6.58.1 Detailed Description

Definition at line 22 of file libxslt.c.

6.58.2 Field Documentation

6.58.2.1 xsltStylesheetPtr xslt_doc::doc

Definition at line 23 of file libxslt.c.

Referenced by free_xsltdoc(), xslt_apply(), xslt_apply_buffer(), and xslt_open().

6.58.2.2 struct bucket_list* xslt_doc::params

Definition at line 24 of file libxslt.c.

Referenced by free_xsltdoc(), xslt_addparam(), xslt_clearparam(), and xslt_open().

The documentation for this struct was generated from the following file:

- src/libxslt.c

6.59 xslt_param Struct Reference

Data Fields

- const char * name
- const char * value

6.59.1 Detailed Description

Definition at line 27 of file libxslt.c.

6.59.2 Field Documentation

6.59.2.1 const char* xslt_param::name

Definition at line 28 of file libxslt.c.

Referenced by free_param(), xslt_addparam(), and xslt_hash().

6.59.2.2 const char* xslt_param::value

Definition at line 29 of file libxslt.c.

Referenced by free_param(), and xslt_addparam().

The documentation for this struct was generated from the following file:

- src/libxslt.c

6.60 zobj Struct Reference

Zlib buffer used for compression and decompression.

`#include <dtsapp.h>`

Data Fields

- uint8_t * buff
  Buffer with compressed/uncompressed data.
- uint16_t olen
  Original size of data.
- uint16_t zlen
  Compressed size of data.

6.60.1 Detailed Description

Zlib buffer used for compression and decompression.

Definition at line 110 of file dtsapp.h.

6.60.2 Field Documentation

6.60.2.1 uint8_t* zobj::buff

Buffer with compressed/uncompressed data.

Definition at line 112 of file dtsapp.h.

Referenced by zcompress(), and zuncompress().

6.60.2.2 uint16_t zobj::olen

Original size of data.

Definition at line 114 of file dtsapp.h.

Referenced by zcompress(), and zuncompress().

6.60.2.3 uint16_t zobj::zlen

Compressed size of data.

Definition at line 116 of file dtsapp.h.

Referenced by zcompress(), and zuncompress().

The documentation for this struct was generated from the following file:

- src/include/dtsapp.h

Chapter 7

File Documentation

7.1 build/config.h File Reference

Macros

- #define HAVE_ARPA_INET_H 1
- #define HAVE_ARPA_NAMESER_H 1
- #define HAVE_CHOWN 1
- #define HAVE_DLFCN_H 1
- #define HAVE_FCNTL_H 1
- #define HAVE_FORK 1
- #define HAVE_GETHOSTBYADDR 1
- #define HAVE_GETPAGESIZE 1
- #define HAVE_GETTIMEOFDAY 1
- #define HAVE_INET_NTOA 1
- #define HAVE_INTTYPES_H 1
- #define HAVE_LIBCRYPTO 1
- #define HAVE_LIBCURL 1
- #define HAVE_LIBM 1
- #define HAVE_LIBNETFILTER_CONNTRACK 1
- #define HAVE_LIBNETFILTER_QUEUE 1
- #define HAVE_LIBPTHREAD 1
- #define HAVE_LIBSSL 1
- #define HAVE_LIBUUID 1
- #define HAVE_LIBZ 1
- #define HAVE_LINUX_IP_H 1
- #define HAVE_LINUX_UN_H 1
- #define HAVE_LINUX_VERSION_H 1
- #define HAVE_MALLOC 1
- #define HAVE_MEMORY_H 1
- #define HAVE_MEMSET 1
- #define HAVE_MMAP 1
- #define HAVE_MUNMAP 1
- #define HAVE_NETDB_H 1
- #define HAVE_NETINET_IN_H 1
- #define HAVE_REALLOC 1
- #define HAVE_RESOLV_H 1
- #define HAVE_SELECT 1
- #define HAVE_SIGNAL_H 1
- #define HAVE_SOCKET 1

- #define HAVE_STDINT_H 1
- #define HAVE_STDLIB_H 1
- #define HAVE_STRCASECMP 1
- #define HAVE_STRCHR 1
- #define HAVE_STRDUP 1
- #define HAVE_STRERROR 1
- #define HAVE_STRINGS_H 1
- #define HAVE_STRING_H 1
- #define HAVE_STRRCHR 1
- #define HAVE_STRSTR 1
- #define HAVE_STRTOL 1
- #define HAVE_STRTOUL 1
- #define HAVE_STRTOULL 1
- #define HAVE_SYSLOG_H 1
- #define HAVE_SYS_FILE_H 1
- #define HAVE_SYS_IOCTL_H 1
- #define HAVE_SYS_PARAM_H 1
- #define HAVE_SYS_SOCKET_H 1
- #define HAVE_SYS_STAT_H 1
- #define HAVE_SYS_TIME_H 1
- #define HAVE_SYS_TYPES_H 1
- #define HAVE_UNISTD_H 1
- #define HAVE_VFORK 1
- #define HAVE_WORKING_FORK 1
- #define HAVE_WORKING_VFORK 1
- #define LIBCURL_FEATURE_ASYNCHDNS 1
- #define LIBCURL_FEATURE_IDN 1
- #define LIBCURL_FEATURE_IPV6 1
- #define LIBCURL_FEATURE_LIBZ 1
- #define LIBCURL_FEATURE_NTLM 1
- #define LIBCURL_FEATURE_SSL 1
- #define LIBCURL_PROTOCOL_DICT 1
- #define LIBCURL_PROTOCOL_FILE 1
- #define LIBCURL_PROTOCOL_FTP 1
- #define LIBCURL_PROTOCOL_FTPS 1
- #define LIBCURL_PROTOCOL_HTTP 1
- #define LIBCURL_PROTOCOL_HTTPS 1
- #define LIBCURL_PROTOCOL_IMAP 1
- #define LIBCURL_PROTOCOL_LDAP 1
- #define LIBCURL_PROTOCOL_POP3 1
- #define LIBCURL_PROTOCOL_RTSP 1
- #define LIBCURL_PROTOCOL_SMTP 1
- #define LIBCURL_PROTOCOL_TELNET 1
- #define LIBCURL_PROTOCOL_TFTP 1
- #define LT_OBJDIR ".libs/"
- #define PACKAGE "dtsapplib"
- #define PACKAGE_BUGREPORT "gregory@distrotech.co.za"
- #define PACKAGE_NAME "dtsapplib"
- #define PACKAGE_STRING "dtsapplib 0.2"
- #define PACKAGE_TARNAME "dtsapplib"
- #define PACKAGE_URL ""
- #define PACKAGE_VERSION "0.2"
- #define STDC_HEADERS 1
- #define VERSION "0.2"

7.1.1 Macro Definition Documentation

7.1.1.1 #define HAVE_ARPA_INET_H 1

Definition at line 5 of file config.h.

7.1.1.2 #define HAVE_ARPA_NAMESER_H 1

Definition at line 8 of file config.h.

7.1.1.3 #define HAVE_CHOWN 1

Definition at line 11 of file config.h.

7.1.1.4 #define HAVE_DLFCN_H 1

Definition at line 14 of file config.h.

7.1.1.5 #define HAVE_FCNTL_H 1

Definition at line 17 of file config.h.

7.1.1.6 #define HAVE_FORK 1

Definition at line 20 of file config.h.

7.1.1.7 #define HAVE_GETHOSTBYADDR 1

Definition at line 23 of file config.h.

7.1.1.8 #define HAVE_GETPAGESIZE 1

Definition at line 26 of file config.h.

7.1.1.9 #define HAVE_GETTIMEOFDAY 1

Definition at line 29 of file config.h.

7.1.1.10 #define HAVE_INET_NTOA 1

Definition at line 32 of file config.h.

7.1.1.11 #define HAVE_INTTYPES_H 1

Definition at line 35 of file config.h.

7.1.1.12 #define HAVE_LIBCRYPTO 1

Definition at line 38 of file config.h.

7.1.1.13 #define HAVE_LIBCURL 1

Definition at line 41 of file config.h.

7.1.1.14 #define HAVE_LIBM 1

Definition at line 44 of file config.h.

7.1.1.15 #define HAVE_LIBNETFILTER_CONNTRACK 1

Definition at line 48 of file config.h.

7.1.1.16 #define HAVE_LIBNETFILTER_QUEUE 1

Definition at line 52 of file config.h.

7.1.1.17 #define HAVE_LIBPTHREAD 1

Definition at line 55 of file config.h.

7.1.1.18 #define HAVE_LIBSSL 1

Definition at line 58 of file config.h.

7.1.1.19 #define HAVE_LIBUUID 1

Definition at line 61 of file config.h.

7.1.1.20 #define HAVE_LIBZ 1

Definition at line 64 of file config.h.

7.1.1.21 #define HAVE_LINUX_IP_H 1

Definition at line 67 of file config.h.

7.1.1.22 #define HAVE_LINUX_UN_H 1

Definition at line 70 of file config.h.

7.1.1.23 #define HAVE_LINUX_VERSION_H 1

Definition at line 73 of file config.h.

7.1.1.24 #define HAVE_MALLOC 1

Definition at line 77 of file config.h.

7.1.1.25 #define HAVE_MEMORY_H 1

Definition at line 80 of file config.h.

7.1.1.26 #define HAVE_MEMSET 1

Definition at line 83 of file config.h.

7.1.1.27 #define HAVE_MMAP 1

Definition at line 86 of file config.h.

7.1.1.28 #define HAVE_MUNMAP 1

Definition at line 89 of file config.h.

7.1.1.29 #define HAVE_NETDB_H 1

Definition at line 92 of file config.h.

7.1.1.30 #define HAVE_NETINET_IN_H 1

Definition at line 95 of file config.h.

7.1.1.31 #define HAVE_REALLOC 1

Definition at line 99 of file config.h.

7.1.1.32 #define HAVE_RESOLV_H 1

Definition at line 102 of file config.h.

7.1.1.33 #define HAVE_SELECT 1

Definition at line 105 of file config.h.

7.1.1.34 #define HAVE_SIGNAL_H 1

Definition at line 108 of file config.h.

7.1.1.35 #define HAVE_SOCKET 1

Definition at line 111 of file config.h.

7.1.1.36 #define HAVE_STDINT_H 1

Definition at line 114 of file config.h.

7.1.1.37 #define HAVE_STDLIB_H 1

Definition at line 117 of file config.h.

7.1.1.38 #define HAVE_STRCASECMP 1

Definition at line 120 of file config.h.

7.1.1.39 #define HAVE_STRCHR 1

Definition at line 123 of file config.h.

7.1.1.40 #define HAVE_STRDUP 1

Definition at line 126 of file config.h.

7.1.1.41 #define HAVE_STRERROR 1

Definition at line 129 of file config.h.

7.1.1.42 #define HAVE_STRING_H 1

Definition at line 135 of file config.h.

7.1.1.43 #define HAVE_STRINGS_H 1

Definition at line 132 of file config.h.

7.1.1.44 #define HAVE_STRRCHR 1

Definition at line 138 of file config.h.

7.1.1.45 #define HAVE_STRSTR 1

Definition at line 141 of file config.h.

7.1.1.46 #define HAVE_STRTOL 1

Definition at line 144 of file config.h.

7.1.1.47 #define HAVE_STRTOUL 1

Definition at line 147 of file config.h.

7.1.1.48 #define HAVE_STRTOULL 1

Definition at line 150 of file config.h.

7.1.1.49 #define HAVE_SYS_FILE_H 1

Definition at line 156 of file config.h.

7.1.1.50 #define HAVE_SYS_IOCTL_H 1

Definition at line 159 of file config.h.

7.1.1.51 #define HAVE_SYS_PARAM_H 1

Definition at line 162 of file config.h.

7.1.1.52 #define HAVE_SYS_SOCKET_H 1

Definition at line 165 of file config.h.

7.1.1.53 #define HAVE_SYS_STAT_H 1

Definition at line 168 of file config.h.

7.1.1.54 #define HAVE_SYS_TIME_H 1

Definition at line 171 of file config.h.

7.1.1.55 #define HAVE_SYS_TYPES_H 1

Definition at line 174 of file config.h.

7.1.1.56 #define HAVE_SYSLOG_H 1

Definition at line 153 of file config.h.

7.1.1.57 #define HAVE_UNISTD_H 1

Definition at line 177 of file config.h.

7.1.1.58 #define HAVE_VFORK 1

Definition at line 180 of file config.h.

7.1.1.59 #define HAVE_WORKING_FORK 1

Definition at line 186 of file config.h.

7.1.1.60 #define HAVE_WORKING_VFORK 1

Definition at line 189 of file config.h.

7.1.1.61 #define LIBCURL_FEATURE_ASYNCHDNS 1

Definition at line 192 of file config.h.

7.1.1.62 #define LIBCURL_FEATURE_IDN 1

Definition at line 195 of file config.h.

7.1.1.63 #define LIBCURL_FEATURE_IPV6 1

Definition at line 198 of file config.h.

7.1.1.64 #define LIBCURL_FEATURE_LIBZ 1

Definition at line 204 of file config.h.

7.1.1.65 #define LIBCURL_FEATURE_NTLM 1

Definition at line 207 of file config.h.

7.1.1.66 #define LIBCURL_FEATURE_SSL 1

Definition at line 210 of file config.h.

7.1.1.67 #define LIBCURL_PROTOCOL_DICT 1

Definition at line 216 of file config.h.

7.1.1.68 #define LIBCURL_PROTOCOL_FILE 1

Definition at line 219 of file config.h.

7.1.1.69 #define LIBCURL_PROTOCOL_FTP 1

Definition at line 222 of file config.h.

7.1.1.70 #define LIBCURL_PROTOCOL_FTPS 1

Definition at line 225 of file config.h.

7.1.1.71 #define LIBCURL_PROTOCOL_HTTP 1

Definition at line 228 of file config.h.

7.1.1.72 #define LIBCURL_PROTOCOL_HTTPS 1

Definition at line 231 of file config.h.

7.1.1.73 #define LIBCURL_PROTOCOL_IMAP 1

Definition at line 234 of file config.h.

7.1.1.74 #define LIBCURL_PROTOCOL_LDAP 1

Definition at line 237 of file config.h.

7.1.1.75 #define LIBCURL_PROTOCOL_POP3 1

Definition at line 240 of file config.h.

7.1.1.76 #define LIBCURL_PROTOCOL_RTSP 1

Definition at line 243 of file config.h.

7.1.1.77 #define LIBCURL_PROTOCOL_SMTP 1

Definition at line 246 of file config.h.

7.1.1.78 #define LIBCURL_PROTOCOL_TELNET 1

Definition at line 249 of file config.h.

7.1.1.79 #define LIBCURL_PROTOCOL_TFTP 1

Definition at line 252 of file config.h.

7.1.1.80 #define LT_OBJDIR ".libs/"

Definition at line 256 of file config.h.

7.1.1.81 #define PACKAGE "dtsapplib"

Definition at line 262 of file config.h.

7.1.1.82 #define PACKAGE_BUGREPORT "gregory@distrotech.co.za"

Definition at line 265 of file config.h.

7.1.1.83 #define PACKAGE_NAME "dtsapplib"

Definition at line 268 of file config.h.

7.1.1.84 #define PACKAGE_STRING "dtsapplib 0.2"

Definition at line 271 of file config.h.

7.1.1.85 #define PACKAGE_TARNAME "dtsapplib"

Definition at line 274 of file config.h.

7.1.1.86 #define PACKAGE_URL ""

Definition at line 277 of file config.h.

7.1.1.87 #define PACKAGE_VERSION "0.2"

Definition at line 280 of file config.h.

7.1.1.88 #define STDC_HEADERS 1

Definition at line 283 of file config.h.

7.1.1.89 #define VERSION "0.2"

Definition at line 286 of file config.h.

7.2 mingw/config.h File Reference

Macros

- #define HAVE_DLFCN_H 1
- #define HAVE_FCNTL_H 1
- #define HAVE_GETPAGESIZE 1
- #define HAVE_GETTIMEOFDAY 1
- #define HAVE_INTTYPES_H 1
- #define HAVE_LIBCRYPTO 1
- #define HAVE_LIBCURL 1
- #define HAVE_LIBM 1
- #define HAVE_LIBPTHREAD 1
- #define HAVE_LIBSSL 1
- #define HAVE_LIBZ 1
- #define HAVE_MALLOC 0
- #define HAVE_MEMORY_H 1
- #define HAVE_MEMSET 1
- #define HAVE_REALLOC 0
- #define HAVE_SIGNAL_H 1
- #define HAVE_STDINT_H 1
- #define HAVE_STDLIB_H 1
- #define HAVE_STRCASECMP 1
- #define HAVE_STRCHR 1
- #define HAVE_STRDUP 1
- #define HAVE_STRERROR 1
- #define HAVE_STRINGS_H 1
- #define HAVE_STRING_H 1
- #define HAVE_STRRCHR 1
- #define HAVE_STRSTR 1
- #define HAVE_STRTOL 1

- #define HAVE_STRTOUL 1
- #define HAVE_STRTOULL 1
- #define HAVE_SYS_FILE_H 1
- #define HAVE_SYS_PARAM_H 1
- #define HAVE_SYS_STAT_H 1
- #define HAVE_SYS_TIME_H 1
- #define HAVE_SYS_TYPES_H 1
- #define HAVE_UNISTD_H 1
- #define LIBCURL_FEATURE_ASYNCHDNS 1
- #define LIBCURL_FEATURE_IDN 1
- #define LIBCURL_FEATURE_IPV6 1
- #define LIBCURL_FEATURE_LIBZ 1
- #define LIBCURL_FEATURE_NTLM 1
- #define LIBCURL_FEATURE_SSL 1
- #define LIBCURL_PROTOCOL_DICT 1
- #define LIBCURL_PROTOCOL_FILE 1
- #define LIBCURL_PROTOCOL_FTP 1
- #define LIBCURL_PROTOCOL_FTPS 1
- #define LIBCURL_PROTOCOL_HTTP 1
- #define LIBCURL_PROTOCOL_HTTPS 1
- #define LIBCURL_PROTOCOL_IMAP 1
- #define LIBCURL_PROTOCOL_LDAP 1
- #define LIBCURL_PROTOCOL_POP3 1
- #define LIBCURL_PROTOCOL_RTSP 1
- #define LIBCURL_PROTOCOL_SMTP 1
- #define LIBCURL_PROTOCOL_TELNET 1
- #define LIBCURL_PROTOCOL_TFTP 1
- #define LT_OBJDIR ".libs/"
- #define PACKAGE "dtsapplib"
- #define PACKAGE_BUGREPORT "gregory@distrotech.co.za"
- #define PACKAGE_NAME "dtsapplib"
- #define PACKAGE_STRING "dtsapplib 0.2"
- #define PACKAGE_TARNAME "dtsapplib"
- #define PACKAGE_URL ""
- #define PACKAGE_VERSION "0.2"
- #define STDC_HEADERS 1
- #define VERSION "0.2"
- #define gid_t int
- #define malloc rpl_malloc
- #define realloc rpl_realloc
- #define uid_t int
- #define vfork fork

7.2.1 Macro Definition Documentation

7.2.1.1 #define gid_t int

Definition at line 307 of file config.h.

7.2.1.2 #define HAVE_DLFCN_H 1

Definition at line 14 of file config.h.

7.2.1.3 #define HAVE_FCNTL_H 1

Definition at line 17 of file config.h.

7.2.1.4 #define HAVE_GETPAGESIZE 1

Definition at line 26 of file config.h.

7.2.1.5 #define HAVE_GETTIMEOFDAY 1

Definition at line 29 of file config.h.

7.2.1.6 #define HAVE_INTTYPES_H 1

Definition at line 35 of file config.h.

7.2.1.7 #define HAVE_LIBCRYPTO 1

Definition at line 38 of file config.h.

7.2.1.8 #define HAVE_LIBCURL 1

Definition at line 41 of file config.h.

7.2.1.9 #define HAVE_LIBM 1

Definition at line 44 of file config.h.

7.2.1.10 #define HAVE_LIBPTHREAD 1

Definition at line 55 of file config.h.

7.2.1.11 #define HAVE_LIBSSL 1

Definition at line 58 of file config.h.

7.2.1.12 #define HAVE_LIBZ 1

Definition at line 64 of file config.h.

7.2.1.13 #define HAVE_MALLOC 0

Definition at line 77 of file config.h.

7.2.1.14 #define HAVE_MEMORY_H 1

Definition at line 80 of file config.h.

7.2.1.15 #define HAVE_MEMSET 1

Definition at line 83 of file config.h.

7.2.1.16 #define HAVE_REALLOC 0

Definition at line 99 of file config.h.

7.2.1.17 #define HAVE_SIGNAL_H 1

Definition at line 108 of file config.h.

7.2.1.18 #define HAVE_STDINT_H 1

Definition at line 114 of file config.h.

7.2.1.19 #define HAVE_STDLIB_H 1

Definition at line 117 of file config.h.

7.2.1.20 #define HAVE_STRCASECMP 1

Definition at line 120 of file config.h.

7.2.1.21 #define HAVE_STRCHR 1

Definition at line 123 of file config.h.

7.2.1.22 #define HAVE_STRDUP 1

Definition at line 126 of file config.h.

7.2.1.23 #define HAVE_STRERROR 1

Definition at line 129 of file config.h.

7.2.1.24 #define HAVE_STRING_H 1

Definition at line 135 of file config.h.

7.2.1.25 #define HAVE_STRINGS_H 1

Definition at line 132 of file config.h.

7.2.1.26 #define HAVE_STRRCHR 1

Definition at line 138 of file config.h.

7.2.1.27 #define HAVE_STRSTR 1

Definition at line 141 of file config.h.

7.2.1.28 #define HAVE_STRTOL 1

Definition at line 144 of file config.h.

7.2.1.29 #define HAVE_STRTOUL 1

Definition at line 147 of file config.h.

7.2.1.30 #define HAVE_STRTOULL 1

Definition at line 150 of file config.h.

7.2.1.31 #define HAVE_SYS_FILE_H 1

Definition at line 156 of file config.h.

7.2.1.32 #define HAVE_SYS_PARAM_H 1

Definition at line 162 of file config.h.

7.2.1.33 #define HAVE_SYS_STAT_H 1

Definition at line 168 of file config.h.

7.2.1.34 #define HAVE_SYS_TIME_H 1

Definition at line 171 of file config.h.

7.2.1.35 #define HAVE_SYS_TYPES_H 1

Definition at line 174 of file config.h.

7.2.1.36 #define HAVE_UNISTD_H 1

Definition at line 177 of file config.h.

7.2.1.37 #define LIBCURL_FEATURE_ASYNCHDNS 1

Definition at line 192 of file config.h.

7.2.1.38 #define LIBCURL_FEATURE_IDN 1

Definition at line 195 of file config.h.

7.2.1.39 #define LIBCURL_FEATURE_IPV6 1

Definition at line 198 of file config.h.

7.2.1.40 #define LIBCURL_FEATURE_LIBZ 1

Definition at line 204 of file config.h.

7.2.1.41 #define LIBCURL_FEATURE_NTLM 1

Definition at line 207 of file config.h.

7.2.1.42 #define LIBCURL_FEATURE_SSL 1

Definition at line 210 of file config.h.

7.2.1.43 #define LIBCURL_PROTOCOL_DICT 1

Definition at line 216 of file config.h.

7.2.1.44 #define LIBCURL_PROTOCOL_FILE 1

Definition at line 219 of file config.h.

7.2.1.45 #define LIBCURL_PROTOCOL_FTP 1

Definition at line 222 of file config.h.

7.2.1.46 #define LIBCURL_PROTOCOL_FTPS 1

Definition at line 225 of file config.h.

7.2.1.47 #define LIBCURL_PROTOCOL_HTTP 1

Definition at line 228 of file config.h.

7.2.1.48 #define LIBCURL_PROTOCOL_HTTPS 1

Definition at line 231 of file config.h.

7.2.1.49 #define LIBCURL_PROTOCOL_IMAP 1

Definition at line 234 of file config.h.

7.2.1.50 #define LIBCURL_PROTOCOL_LDAP 1

Definition at line 237 of file config.h.

7.2.1.51 #define LIBCURL_PROTOCOL_POP3 1

Definition at line 240 of file config.h.

7.2.1.52 #define LIBCURL_PROTOCOL_RTSP 1

Definition at line 243 of file config.h.

7.2.1.53 #define LIBCURL_PROTOCOL_SMTP 1

Definition at line 246 of file config.h.

7.2.1.54 #define LIBCURL_PROTOCOL_TELNET 1

Definition at line 249 of file config.h.

7.2.1.55 #define LIBCURL_PROTOCOL_TFTP 1

Definition at line 252 of file config.h.

7.2.1.56 #define LT_OBJDIR ".libs/"

Definition at line 256 of file config.h.

7.2.1.57 #define malloc rpl_malloc

Definition at line 316 of file config.h.

Referenced by addtobucket(), create_bucketlist(), framework_mkcore(), ipv6to4prefix(), ldap_search_base(), ldap_search_one(), ldap_search_sub(), ldap_simplebind(), ldap_simplerebind(), ll_remember_index(), new_radpacket(), objalloc(), sslstartup(), xml_createpath(), xslt_addparam(), and zcompress().

7.2.1.58 #define PACKAGE "dtsapplib"

Definition at line 262 of file config.h.

7.2.1.59 #define PACKAGE_BUGREPORT "gregory@distrotech.co.za"

Definition at line 265 of file config.h.

7.2.1.60 #define PACKAGE_NAME "dtsapplib"

Definition at line 268 of file config.h.

7.2.1.61 #define PACKAGE_STRING "dtsapplib 0.2"

Definition at line 271 of file config.h.

7.2.1.62 #define PACKAGE_TARNAME "dtsapplib"

Definition at line 274 of file config.h.

7.2.1.63 #define PACKAGE_URL ""

Definition at line 277 of file config.h.

7.2.1.64 #define PACKAGE_VERSION "0.2"

Definition at line 280 of file config.h.

7.2.1.65 #define realloc rpl_realloc

Definition at line 322 of file config.h.

Referenced by getcmdline(), and gzinflatebuf().

7.2.1.66 #define STDC_HEADERS 1

Definition at line 283 of file config.h.

7.2.1.67 #define uid_t int

Definition at line 331 of file config.h.

7.2.1.68 #define VERSION "0.2"

Definition at line 286 of file config.h.

7.2.1.69 #define vfork fork

Definition at line 350 of file config.h.

7.3 src/config.c File Reference

INI style config file interface.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include "include/dtsapp.h"
```

Data Structures

- struct config_category
- struct config_file

Functions

- void initconfigfiles (void)
- void unrefconfigfiles (void)
- int process_config (const char *configname, const char *configfile)

- struct bucket_list * get_config_file (const char *configname)
- struct bucket_list * get_config_category (const char *configname, const char *category)
- struct bucket_list * get_category_next (struct bucket_loop *cloop, char *name, int len)
- struct bucket_loop * get_category_loop (const char *configname)
- void config_entry_callback (struct bucket_list *entries, config_entrycb entry_cb)
- void config_cat_callback (struct bucket_list *categories, config_catcb cat_cb)
- void config_file_callback (config_filecb file_cb)
- struct config_entry * get_config_entry (struct bucket_list *categories, const char *item)

7.3.1 Detailed Description

INI style config file interface.

Definition in file config.c.

7.4 src/curl.c File Reference

CURL Interface.

```
#include <string.h>
#include <stdint.h>
#include <stdlib.h>
#include <curl/curl.h>
#include <curl/easy.h>
#include "dtsapp.h"
```

Data Structures

- struct **curl_progress**
- struct **curl_password**
- struct curl_post

Functions

- int curlinit (void)
- void curlclose (void)
- struct curlbuf * curl_geturl (const char *def_url, struct basic_auth *bauth, curl_authcb authcb, void *auth_data)
- struct curlbuf * curl_posturl (const char *def_url, struct basic_auth *bauth, struct curl_post *post, curl_authcb authcb, void *auth_data)
- struct curlbuf * curl_ungzip (struct curlbuf *cbuf)
- struct basic_auth * curl_newauth (const char *user, const char *passwd)
- void free_post (void *data)
- struct curl_post * curl_newpost (void)
- void curl_postitem (struct curl_post *post, const char *name, const char *item)
- char * url_escape (char *url)
- char * url_unescape (char *url)
- void free_progress (void *data)
- void curl_setprogress (curl_progress_func cb, curl_progress_pause p_cb, curl_progress_newdata d_cb, void *data)
- void free_curlpassword (void *data)
- void curl_setauth_cb (curl_authcb auth_cb, void *data)
- struct xml_doc * curl_buf2xml (struct curlbuf *cbuf)

7.4.1 Detailed Description

CURL Interface.

Definition in file curl.c.

7.5 src/fileutil.c File Reference

File utilities to test files (fstat)

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <fcntl.h>
#include <ctype.h>
#include <grp.h>
```

Functions

- int is_file (const char *path)
- int is_dir (const char *path)
- int is_exec (const char *path)
- int mk_dir (const char *dir, mode_t mode, uid_t user, gid_t group)

7.5.1 Detailed Description

File utilities to test files (fstat)

Definition in file fileutil.c.

7.6 src/include/dtsapp.h File Reference

DTS Application library API Include file.

```
#include <signal.h>
#include <arpa/inet.h>
```

Data Structures

- union sockstruct
- struct fwsocket
- struct config_entry
- struct zobj

  Zlib buffer used for compression and decompression.
- struct framework_core

  Application framework data.

- struct xml_attr
- struct xml_node
- struct ldap_rdn
- struct ldap_attrval
- struct ldap_attr
- struct ldap_entry
- struct ldap_results
- struct basic_auth
- struct curlbuf

Macros

- #define RAD_AUTH_HDR_LEN 20
- #define RAD_AUTH_PACKET_LEN 4096
- #define RAD_AUTH_TOKEN_LEN 16
- #define RAD_MAX_PASS_LEN 128
- #define RAD_ATTR_USER_NAME 1 /∗string∗/
- #define RAD_ATTR_USER_PASSWORD 2 /∗passwd∗/
- #define RAD_ATTR_NAS_IP_ADDR 4 /∗ip∗/
- #define RAD_ATTR_NAS_PORT 5 /∗int∗/
- #define RAD_ATTR_SERVICE_TYPE 6 /∗int∗/
- #define RAD_ATTR_ACCTID 44
- #define RAD_ATTR_PORT_TYPE 61 /∗int∗/
- #define RAD_ATTR_EAP 79 /∗oct∗/
- #define RAD_ATTR_MESSAGE 80 /∗oct∗/
- #define JHASH_INITVAL 0xdeadbeef
- #define jenhash(key, length, initval) hashlittle(key, length, (initval) ? initval : JHASH_INITVAL);

  Define jenhash as hashlittle on big endian it should be hashbig.
- #define clearflag(obj, flag)
- #define setflag(obj, flag)
- #define testflag(obj, flag) (objlock(obj) | (obj->flags & flag) | objunlock(obj))
- #define FRAMEWORK_MAIN(progname, name, email, www, year, runfile, flags, sighfunc)

  A macro to replace main() with initilization and daemonization code.
- #define ALLOC_CONST(const_var, val)

  Macro to assign values to char const.
- #define DTS_OJBREF_CLASS(classtype)

  Add this macro to a C++ class to add refobj support.

Typedefs

- typedef struct ssldata ssldata

  Forward decleration of structure.
- typedef struct natmap natmap

  Forward decleration of structure.
- typedef struct radius_packet radius_packet

  Forward decleration of structure.
- typedef struct nfq_queue nfq_queue

  Forward decleration of structure.
- typedef struct nfq_data nfq_data

  Forward decleration of structure.
- typedef struct nfct_struct nfct_struct

  Forward decleration of structure.

- typedef struct nfqnl_msg_packet_hdr nfqnl_msg_packet_hdr

  Forward decleration of structure.
- typedef void(* radius_cb)(struct radius_packet *, void *)
- typedef int(* frameworkfunc)(int, char **)

  Framework callback function.
- typedef void(* syssighandler)(int, siginfo_t *, void *)

  Callback to user supplied signal handler.
- typedef void(* threadcleanup)(void *)

  Function called after thread termination.
- typedef void *(* threadfunc)(void **)

  Thread function.
- typedef int(* threadsighandler)(int, void *)

  Thread signal handler function.
- typedef int(* blisthash)(const void *, int)
- typedef void(* objdestroy)(void *)
- typedef void(* socketrecv)(struct fwsocket *, void *)
- typedef void(* blist_cb)(void *, void *)
- typedef void(* config_filecb)(struct bucket_list *, const char *, const char *)
- typedef void(* config_catcb)(struct bucket_list *, const char *)
- typedef void(* config_entrycb)(const char *, const char *)
- typedef uint32_t(* nfqueue_cb)(struct nfq_data *, struct nfqnl_msg_packet_hdr *, char *, uint32_t, void *, uint32_t *, void **)
- typedef struct xml_node xml_node

  Forward decleration of structure.
- typedef struct xml_search xml_search

  Forward decleration of structure.
- typedef struct xml_doc xml_doc

  Forward decleration of structure.
- typedef struct xslt_doc xslt_doc

  Forward decleration of structure.
- typedef struct ldap_conn ldap_conn

  Forward decleration of structure.
- typedef struct ldap_modify ldap_modify

  Forward decleration of structure.
- typedef struct ldap_add ldap_add

  Forward decleration of structure.
- typedef struct curl_post curl_post

  Forward decleration of structure.
- typedef struct basic_auth *(* curl_authcb)(const char *user, const char *passwd, void *data)
- typedef int(* curl_progress_func)(void *, double, double, double, double)
- typedef void(* curl_progress_pause)(void *, int)
- typedef void *(* curl_progress_newdata)(void *)

Enumerations

- enum sock_flags { SOCK_FLAG_BIND = 1 << 0, SOCK_FLAG_CLOSE = 1 << 1 }
- enum framework_flags { FRAMEWORK_FLAG_DAEMON = 1 << 0, FRAMEWORK_FLAG_NOGNU = 1 << 1, FRAMEWORK_FLAG_NOTHREAD = 1 << 2 }

  Application control flags.
- enum RADIUS_CODE {
  RAD_CODE_AUTHREQUEST = 1, RAD_CODE_AUTHACCEPT = 2, RAD_CODE_AUTHREJECT = 3, RAD_CODE_ACCTREQUEST = 4,
  RAD_CODE_ACCTRESPONSE = 5, RAD_CODE_AUTHCHALLENGE = 11 }

- enum ldap_starttls { LDAP_STARTTLS_NONE, LDAP_STARTTLS_ATTEMPT, LDAP_STARTTLS_ENFORCE }
- enum ldap_attrtype { LDAP_ATTRTYPE_CHAR, LDAP_ATTRTYPE_B64, LDAP_ATTRTYPE_OCTET }

Functions

- void framework_mkcore (char *progname, char *name, char *email, char *web, int year, char *runfile, int flags, syssighandler sigfunc)

  Initilise application data structure and return a reference.
- int framework_init (int argc, char *argv[], frameworkfunc callback)

  Initilise the application daemonise and join the manager thread.
- void printgnu ()

  Print a brief GNU copyright notice on console.
- void daemonize ()

  Daemonise the application using fork/exit.
- int lockpidfile (const char *runfile)

  Lock the run file in the framework application info.
- struct thread_pvt * framework_mkthread (threadfunc, threadcleanup, threadsighandler, void *data)

  create a thread result must be unreferenced
- void framework_unixsocket (char *sock, int protocol, int mask, threadfunc connectfunc, threadcleanup cleanup)

  Create and run UNIX socket thread.
- int framework_threadok (void *data)

  let threads check there status by passing in a pointer to there data
- int startthreads (void)

  initialise the threadlist start manager thread
- void stopthreads (void)

  Stoping the manager thread will stop all other threads.
- int objlock (void *data)
- int objtrylock (void *data)
- int objunlock (void *data)
- int objcnt (void *data)
- int objsize (void *data)
- int objunref (void *data)
- int objref (void *data)
- void * objalloc (int size, objdestroy)
- void * objchar (const char *orig)
- void * create_bucketlist (int bitmask, blisthash hash_function)
- int addtobucket (struct bucket_list *blist, void *data)
- void remove_bucket_item (struct bucket_list *blist, void *data)
- int bucket_list_cnt (struct bucket_list *blist)
- void * bucket_list_find_key (struct bucket_list *list, const void *key)
- void bucketlist_callback (struct bucket_list *blist, blist_cb callback, void *data2)
- struct bucket_loop * init_bucket_loop (struct bucket_list *blist)
- void stop_bucket_loop (struct bucket_loop *bloop)
- void * next_bucket_loop (struct bucket_loop *bloop)
- void remove_bucket_loop (struct bucket_loop *bloop)
- uint32_t hashlittle (const void *key, size_t length, uint32_t initval)

  hash a variable-length key into a 32-bit value (Little Endian)
- void seedrand (void)

  Seed openssl random number generator.
- int genrand (void *buf, int len)

Generate random sequence.

- void sha512sum (unsigned char *buff, const void *data, unsigned long len)

  Calculate the SHA2-512 hash.
- void sha256sum (unsigned char *buff, const void *data, unsigned long len)

  Calculate the SHA2-256 hash.
- void sha1sum (unsigned char *buff, const void *data, unsigned long len)

  Calculate the SHA1 hash.
- void md5sum (unsigned char *buff, const void *data, unsigned long len)

  Calculate the MD5 hash.
- void sha512sum2 (unsigned char *buff, const void *data, unsigned long len, const void *data2, unsigned long len2)

  Calculate the SHA2-512 hash accross 2 data chunks.
- void sha256sum2 (unsigned char *buff, const void *data, unsigned long len, const void *data2, unsigned long len2)

  Calculate the SHA2-256 hash accross 2 data chunks.
- void sha1sum2 (unsigned char *buff, const void *data, unsigned long len, const void *data2, unsigned long len2)

  Calculate the SHA1 hash accross 2 data chunks.
- void md5sum2 (unsigned char *buff, const void *data, unsigned long len, const void *data2, unsigned long len2)

  Calculate the MD5 hash accross 2 data chunks.
- int sha512cmp (unsigned char *digest1, unsigned char *digest2)

  Compare two SHA2-512 hashes.
- int sha256cmp (unsigned char *digest1, unsigned char *digest2)

  Compare two SHA2-256 hashes.
- int sha1cmp (unsigned char *digest1, unsigned char *digest2)

  Compare two SHA1 hashes.
- int md5cmp (unsigned char *digest1, unsigned char *digest2)

  Compare two md5 hashes.
- void sha512hmac (unsigned char *buff, const void *data, unsigned long len, const void *key, unsigned long klen)

  Hash Message Authentication Codes (HMAC) SHA2-512.
- void sha256hmac (unsigned char *buff, const void *data, unsigned long len, const void *key, unsigned long klen)

  Hash Message Authentication Codes (HMAC) SHA2-256.
- void sha1hmac (unsigned char *buff, const void *data, unsigned long len, const void *key, unsigned long klen)

  Hash Message Authentication Codes (HMAC) SHA1.
- void md5hmac (unsigned char *buff, const void *data, unsigned long len, const void *key, unsigned long klen)

  Hash Message Authentication Codes (HMAC) MD5.
- int strlenzero (const char *str)

  Check if a string is zero length.
- char * ltrim (char *str)

  Trim white space at the begining of a string.
- char * rtrim (const char *str)

  Trim white space at the end of a string.
- char * trim (const char *str)

  Trim whitesapce from the beggining and end of a string.
- uint64_t tvtontp64 (struct timeval *tv)

  Convert a timeval struct to 64bit NTP time.
- uint16_t checksum (const void *data, int len)

Obtain the checksum for a buffer.

- uint16_t checksum_add (const uint16_t checksum, const void *data, int len)

  Obtain the checksum for a buffer adding a checksum.
- uint16_t verifysum (const void *data, int len, const uint16_t check)

  Verify a checksum.
- struct zobj * zcompress (uint8_t *buff, uint16_t len, uint8_t level)

  Allocate a buffer and return it with compressed data.
- void zuncompress (struct zobj *buff, uint8_t *obuff)

  Uncompress zobj buffer to buffer.
- uint8_t * gzinflatebuf (uint8_t *buf_in, int buf_size, uint32_t *len)

  Ungzip a buffer.
- int is_gzip (uint8_t *buf, int buf_size)

  check a buffer if it contains gzip magic
- void touch (const char *filename, uid_t user, gid_t group)

  Create a file and set user and group.
- char * b64enc (const char *message, int nonl)

  Base 64 encode a string.
- char * b64enc_buf (const char *message, uint32_t len, int nonl)

  Base 64 encode a buffer.
- struct fwsocket * make_socket (int family, int type, int proto, void *ssl)
- struct fwsocket * sockconnect (int family, int stype, int proto, const char *ipaddr, const char *port, void *ssl)
- struct fwsocket * udpconnect (const char *ipaddr, const char *port, void *ssl)
- struct fwsocket * tcpconnect (const char *ipaddr, const char *port, void *ssl)
- struct fwsocket * sockbind (int family, int stype, int proto, const char *ipaddr, const char *port, void *ssl, int backlog)
- struct fwsocket * udpbind (const char *ipaddr, const char *port, void *ssl)
- struct fwsocket * tcpbind (const char *ipaddr, const char *port, void *ssl, int backlog)
- void close_socket (struct fwsocket *sock)
- void socketclient (struct fwsocket *sock, void *data, socketrecv read, threadcleanup cleanup)
- void socketserver (struct fwsocket *sock, socketrecv connectfunc, socketrecv acceptfunc, threadcleanup cleanup, void *data)
- int checkipv6mask (const char *ipaddr, const char *network, uint8_t bits)
- void ipv4tcpchecksum (uint8_t *pkt)
- void ipv4udpchecksum (uint8_t *pkt)
- void icmpchecksum (uint8_t *pkt)
- void ipv4checksum (uint8_t *pkt)
- int packetchecksumv4 (uint8_t *pkt)
- int packetchecksumv6 (uint8_t *pkt)
- int packetchecksum (uint8_t *pkt)
- void rfc6296_map (struct natmap *map, struct in6_addr *ipaddr, int out)
- int rfc6296_map_add (char *intaddr, char *extaddr)
- const char * cidrtosn (int bitlen, const char *buf, int size)
- const char * getnetaddr (const char *ipaddr, int cidr, const char *buf, int size)
- const char * getbcaddr (const char *ipaddr, int cidr, const char *buf, int size)
- const char * getfirstaddr (const char *ipaddr, int cidr, const char *buf, int size)
- const char * getlastaddr (const char *ipaddr, int cidr, const char *buf, int size)
- uint32_t cidrcnt (int bitlen)
- int reservedip (const char *ipaddr)
- char * ipv6to4prefix (const char *ipaddr)
- int check_ipv4 (const char *ip, int cidr, const char *test)
- struct nfq_queue * nfqueue_attach (uint16_t pf, uint16_t num, uint8_t mode, uint32_t range, nfqueue_cb cb, void *data)

- uint16_t snprintf_pkt (struct nfq_data *tb, struct nfqnl_msg_packet_hdr *ph, uint8_t *pkt, char *buff, uint16_t len)
- struct nf_conntrack * nf_ctrack_buildct (uint8_t *pkt)
- uint8_t nf_ctrack_delete (uint8_t *pkt)
- uint8_t nf_ctrack_nat (uint8_t *pkt, uint32_t addr, uint16_t port, uint8_t dnat)
- void nf_ctrack_dump (void)
- struct nfct_struct * nf_ctrack_trace (void)
- void nf_ctrack_endtrace (struct nfct_struct *nfct)
- uint8_t nf_ctrack_init (void)
- void nf_ctrack_close (void)
- int delete_kernvlan (char *ifname, int vid)
- int create_kernvlan (char *ifname, unsigned short vid)
- int delete_kernmac (char *macdev)
- int create_kernmac (char *ifname, char *macdev, unsigned char *mac)
- int interface_bind (char *iface, int protocol, int flags)
- void randhwaddr (unsigned char *addr)
- int create_tun (const char *ifname, const unsigned char *hwaddr, int flags)
- int ifrename (const char *oldname, const char *newname)
- int ifdown (const char *ifname, int flags)
- int ifup (const char *ifname, int flags)
- int ifhwaddr (const char *ifname, unsigned char *hwaddr)
- int set_interface_flags (int ifindex, int set, int clear)
- int get_iface_index (const char *ifname)
- int set_interface_addr (int ifindex, const unsigned char *hwaddr)
- int set_interface_name (int ifindex, const char *name)
- int set_interface_ipaddr (char *ifname, char *ipaddr)
- int get_ip6_addrprefix (const char *iface, unsigned char *prefix)
- int eui48to64 (unsigned char *mac48, unsigned char *eui64)
- void closenetlink (void)
- unsigned char * addradattr (struct radius_packet *packet, char type, unsigned char *val, char len)
- void addradattrint (struct radius_packet *packet, char type, unsigned int val)
- void addradattrip (struct radius_packet *packet, char type, char *ipaddr)
- void addradattrstr (struct radius_packet *packet, char type, char *str)
- struct radius_packet * new_radpacket (unsigned char code, unsigned char id)
- int send_radpacket (struct radius_packet *packet, const char *userpass, radius_cb read_cb, void *cb_data)
- void add_radserver (const char *ipaddr, const char *auth, const char *acct, const char *secret, int timeout)
- unsigned char * radius_attr_first (struct radius_packet *packet)
- unsigned char * radius_attr_next (struct radius_packet *packet, unsigned char *attr)
- void sslstartup (void)
- void * tlsv1_init (const char *cacert, const char *cert, const char *key, int verify)
- void * sslv2_init (const char *cacert, const char *cert, const char *key, int verify)
- void * sslv3_init (const char *cacert, const char *cert, const char *key, int verify)
- void * dtlsv1_init (const char *cacert, const char *cert, const char *key, int verify)
- int socketread (struct fwsocket *sock, void *buf, int num)
- int socketwrite (struct fwsocket *sock, const void *buf, int num)
- int socketread_d (struct fwsocket *sock, void *buf, int num, union sockstruct *addr)
- int socketwrite_d (struct fwsocket *sock, const void *buf, int num, union sockstruct *addr)
- void ssl_shutdown (void *ssl)
- void tlsaccept (struct fwsocket *sock, struct ssldata *orig)
- struct fwsocket * dtls_listenssl (struct fwsocket *sock)
- void startsslclient (struct fwsocket *sock)
- void initconfigfiles (void)
- void unrefconfigfiles (void)
- int process_config (const char *confignname, const char *configfile)
- struct bucket_loop * get_category_loop (const char *confignname)

- struct bucket_list * get_category_next (struct bucket_loop *cloop, char *name, int len)
- struct bucket_list * get_config_category (const char *configname, const char *category)
- struct config_entry * get_config_entry (struct bucket_list *categories, const char *item)
- void config_file_callback (config_filecb file_cb)
- void config_cat_callback (struct bucket_list *categories, config_catcb entry_cb)
- void config_entry_callback (struct bucket_list *entries, config_entrycb entry_cb)
- struct xml_doc * xml_loaddoc (const char *docfile, int validate)
- struct xml_doc * xml_loadbuf (const uint8_t *buffer, uint32_t len, int validate)
- struct xml_node * xml_getfirstnode (struct xml_search *xpsearch, void **iter)
- struct xml_node * xml_getnextnode (void *iter)
- struct bucket_list * xml_getnodes (struct xml_search *xpsearch)
- struct xml_search * xml_xpath (struct xml_doc *xmldata, const char *xpath, const char *attrkey)
- int xml_nodecount (struct xml_search *xsearch)
- struct xml_node * xml_getnode (struct xml_search *xsearch, const char *key)
- const char * xml_getattr (struct xml_node *xnode, const char *attr)
- void xml_modify (struct xml_doc *xmldoc, struct xml_node *xnode, const char *value)
- void xml_setattr (struct xml_doc *xmldoc, struct xml_node *xnode, const char *name, const char *value)
- struct xml_node * xml_addnode (struct xml_doc *xmldoc, const char *xpath, const char *name, const char *value, const char *attrkey, const char *keyval)
- void xml_appendnode (struct xml_doc *xmldoc, const char *xpath, struct xml_node *child)
- void xml_unlink (struct xml_node *xnode)
- void xml_delete (struct xml_node *xnode)
- char * xml_getbuffer (void *buffer)
- void * xml_doctobuffer (struct xml_doc *xmldoc)
- const char * xml_getrootname (struct xml_doc *xmldoc)
- struct xml_node * xml_getrootnode (struct xml_doc *xmldoc)
- void xml_savefile (struct xml_doc *xmldoc, const char *file, int format, int compress)
- void xml_createpath (struct xml_doc *xmldoc, const char *xpath)
- void xml_init ()
- void xml_close ()
- struct xslt_doc * xslt_open (const char *xsltfile)
- void xslt_addparam (struct xslt_doc *xsltdoc, const char *param, const char *value)
- void xslt_apply (struct xml_doc *xmldoc, struct xslt_doc *xsltdoc, const char *filename, int comp)
- void * xslt_apply_buffer (struct xml_doc *xmldoc, struct xslt_doc *xsltdoc)
- void xslt_init ()
- void xslt_close ()
- struct ldap_conn * ldap_connect (const char *uri, enum ldap_starttls starttls, int timelimit, int limit, int debug, int *err)
- int ldap_simplebind (struct ldap_conn *ld, const char *dn, const char *passwd)
- int ldap_saslbind (struct ldap_conn *ld, const char *mech, const char *realm, const char *authcid, const char *passwd, const char *authzid)
- int ldap_simplerebind (struct ldap_conn *ld, const char *initialdn, const char *initialpw, const char *base, const char *filter, const char *uidrdn, const char *uid, const char *passwd)
- void ldap_close (struct ldap_conn *ld)
- const char * ldap_errmsg (int res)
- struct ldap_results * ldap_search_sub (struct ldap_conn *ld, const char *base, const char *filter, int b64enc, int *res,...)
- struct ldap_results * ldap_search_one (struct ldap_conn *ld, const char *base, const char *filter, int b64enc, int *res,...)
- struct ldap_results * ldap_search_base (struct ldap_conn *ld, const char *base, const char *filter, int b64enc, int *res,...)
- void ldap_unref_entry (struct ldap_results *results, struct ldap_entry *entry)
- void ldap_unref_attr (struct ldap_entry *entry, struct ldap_attr *attr)
- struct ldap_entry * ldap_getentry (struct ldap_results *results, const char *dn)
- struct ldap_attr * ldap_getattr (struct ldap_entry *entry, const char *attr)

- struct ldap_modify * ldap_modifyinit (const char *dn)
- int ldap_mod_del (struct ldap_modify *lmod, const char *attr,...)
- int ldap_mod_add (struct ldap_modify *lmod, const char *attr,...)
- int ldap_mod_rep (struct ldap_modify *lmod, const char *attr,...)
- int ldap_domodify (struct ldap_conn *ld, struct ldap_modify *lmod)
- int ldap_mod_rematttr (struct ldap_conn *ldap, const char *dn, const char *attr)
- int ldap_mod_delattr (struct ldap_conn *ldap, const char *dn, const char *attr, const char *value)
- int ldap_mod_addattr (struct ldap_conn *ldap, const char *dn, const char *attr, const char *value)
- int ldap_mod_repattr (struct ldap_conn *ldap, const char *dn, const char *attr, const char *value)
- int curlinit (void)
- void curlclose (void)
- struct basic_auth * curl_newauth (const char *user, const char *passwd)
- struct curlbuf * curl_geturl (const char *def_url, struct basic_auth *bauth, curl_authcb authcb, void *data)
- void curl_setprogress (curl_progress_func cb, curl_progress_pause p_cb, curl_progress_newdata d_cb, void *data)
- void curl_setauth_cb (curl_authcb auth_cb, void *data)
- struct curl_post * curl_newpost (void)
- void curl_postitem (struct curl_post *post, const char *name, const char *item)
- struct curlbuf * curl_posturl (const char *def_url, struct basic_auth *bauth, struct curl_post *post, curl_authcb authcb, void *data)
- struct curlbuf * curl_ungzip (struct curlbuf *cbuf)
- struct xml_doc * curl_buf2xml (struct curlbuf *cbuf)
- char * url_escape (char *url)
- char * url_unescape (char *url)
- int is_file (const char *path)
- int is_dir (const char *path)
- int is_exec (const char *path)
- int mk_dir (const char *dir, mode_t mode, uid_t user, gid_t group)

7.6.1 Detailed Description

DTS Application library API Include file. The library foremostly implements reference counted objects and hashed bucket lists Referenced Objects these are then used to implement simpler API's to common tasks.

Key components

INI style config file parser.
CURL wraper with support for GET/POST, authentification and progress indication.
File utilities as a wrapper arround fstat.
IP 4/6 Utilities for calculating / checking subnets and checksuming packets.
Interface API for Linux networking including libnetlink from iproute2
XML/XSLT Simplified API for reading, managing and applying transforms.
Some Application shortcuts and wrapper for main quick and dirty daemon app.
Wrappers for Linux netfilter connection tracking and packet queueing
Open LDAP API.
Basic implementation of RADIUS.
Implementation of RFC 6296.
Thread API using pthreads.
Simple implementation of UNIX Domain socket.
Various Utilities including hashing and checksum.
Z Lib Compression/Uncompression Functions.

Definition in file dtsapp.h.

7.6.2 Macro Definition Documentation

7.6.2.1 #define ALLOC_CONST(*const_var, val*)

Value:

```
{ \
        char *tmp_char; \
        if (val) { \
            tmp_char = (char*)malloc(strlen(val) + 1); \
            strcpy(tmp_char, val); \
            const_var = (const char*)tmp_char; \
        } else { \
            const_var = NULL; \
        } \
    }
```

Macro to assign values to char const.

Definition at line 676 of file dtsapp.h.

Referenced by add_modifyval(), add_radserver(), framework_mkcore(), ldap_addinit(), ldap_getent(), ldap_modifyinit(), ldap_saslbind(), new_modreq(), xml_modify(), xml_nodetohash(), and xslt_addparam().

7.6.2.2 #define clearflag(*obj, flag*)

Value:

```
objlock(obj); \
    obj->flags &= ~flag; \
    objunlock(obj)
```

Definition at line 644 of file dtsapp.h.

Referenced by stopthreads().

7.6.2.3 #define DTS_OJBREF_CLASS(*classtype*)

Value:

```
void *operator new(size_t sz) {\
    return objalloc(sz, &classtype::dts_unref_classtype);\
}\
void operator delete(void *obj) {\
}\
static void dts_unref_classtype(void *data) {\
    delete (classtype*)data;\
}\
~classtype()
```

Add this macro to a C++ class to add refobj support.

This macro defines operator overloads for new/delete and declares a destructor.

Note

this should not be used with inheritance

Definition at line 692 of file dtsapp.h.

7.6.2.4 #define RAD_ATTR_ACCTID 44

Definition at line 386 of file dtsapp.h.

7.6.2.5 #define RAD_ATTR_EAP 79 /*oct*/

Definition at line 388 of file dtsapp.h.

7.6.2.6 #define RAD_ATTR_MESSAGE 80 /*oct*/

Definition at line 389 of file dtsapp.h.

7.6.2.7 #define RAD_ATTR_NAS_IP_ADDR 4 /*ip*/

Definition at line 383 of file dtsapp.h.

7.6.2.8 #define RAD_ATTR_NAS_PORT 5 /*int*/

Definition at line 384 of file dtsapp.h.

7.6.2.9 #define RAD_ATTR_PORT_TYPE 61 /*int*/

Definition at line 387 of file dtsapp.h.

7.6.2.10 #define RAD_ATTR_SERVICE_TYPE 6 /*int*/

Definition at line 385 of file dtsapp.h.

7.6.2.11 #define RAD_ATTR_USER_NAME 1 /*string*/

Definition at line 381 of file dtsapp.h.

7.6.2.12 #define RAD_ATTR_USER_PASSWORD 2 /*passwd*/

Definition at line 382 of file dtsapp.h.

7.6.2.13 #define RAD_AUTH_HDR_LEN 20

Definition at line 376 of file dtsapp.h.

Referenced by addradattr(), new_radpacket(), and radius_attr_next().

7.6.2.14 #define RAD_AUTH_PACKET_LEN 4096

Definition at line 377 of file dtsapp.h.

7.6.2.15 #define RAD_AUTH_TOKEN_LEN 16

Definition at line 378 of file dtsapp.h.

Referenced by new_radpacket().

7.6.2.16 #define RAD_MAX_PASS_LEN 128

Definition at line 379 of file dtsapp.h.

7.6.2.17 #define setflag(*obj, flag*)

Value:

```
objlock(obj); \
    obj->flags |= flag; \
    objunlock(obj)
```

Definition at line 648 of file dtsapp.h.

Referenced by close_socket(), nf_ctrack_endtrace(), and socketwrite_d().

7.6.2.18 #define testflag(*obj, flag*) (objlock(obj) | (obj->flags & flag) | objunlock(obj))

Definition at line 652 of file dtsapp.h.

Referenced by framework_threadok().

7.6.3 Typedef Documentation

7.6.3.1 typedef void(* blist_cb)(void *, void *)

Definition at line 173 of file dtsapp.h.

7.6.3.2 typedef int(* blisthash)(const void *, int)

Definition at line 170 of file dtsapp.h.

7.6.3.3 typedef void(* config_catcb)(struct bucket_list *, const char *)

Definition at line 175 of file dtsapp.h.

7.6.3.4 typedef void(* config_entrycb)(const char *, const char *)

Definition at line 176 of file dtsapp.h.

7.6.3.5 typedef void(* config_filecb)(struct bucket_list *, const char *, const char *)

Definition at line 174 of file dtsapp.h.

7.6.3.6 typedef struct basic_auth*(* curl_authcb)(const char *user, const char *passwd, void *data) [read]

Definition at line 603 of file dtsapp.h.

7.6.3.7 typedef struct curl_post curl_post

Forward decleration of structure.

Definition at line 602 of file dtsapp.h.

7.6.3.8 typedef int(* curl_progress_func)(void *, double, double, double, double)

Definition at line 604 of file dtsapp.h.

7.6.3.9 typedef void*(* curl_progress_newdata)(void *)

Definition at line 606 of file dtsapp.h.

7.6.3.10 typedef void(* curl_progress_pause)(void *, int)

Definition at line 605 of file dtsapp.h.

7.6.3.11 typedef struct ldap_add ldap_add

Forward decleration of structure.

Definition at line 553 of file dtsapp.h.

7.6.3.12 typedef struct ldap_conn ldap_conn

Forward decleration of structure.

Definition at line 549 of file dtsapp.h.

7.6.3.13 typedef struct ldap_modify ldap_modify

Forward decleration of structure.

Definition at line 551 of file dtsapp.h.

7.6.3.14 typedef struct natmap natmap

Forward decleration of structure.

Definition at line 120 of file dtsapp.h.

7.6.3.15 typedef struct nfct_struct nfct_struct

Forward decleration of structure.

Definition at line 128 of file dtsapp.h.

7.6.3.16 typedef struct nfq_data nfq_data

Forward decleration of structure.

Definition at line 126 of file dtsapp.h.

7.6.3.17 typedef struct nfq_queue nfq_queue

Forward decleration of structure.

Definition at line 124 of file dtsapp.h.

7.6.3.18 typedef struct nfqnl_msg_packet_hdr nfqnl_msg_packet_hdr

Forward decleration of structure.

Definition at line 130 of file dtsapp.h.

7.6.3.19 typedef uint32_t(* nfqueue_cb)(struct nfq_data *, struct nfqnl_msg_packet_hdr *, char *, uint32_t, void *, uint32_t *, void **)

Definition at line 177 of file dtsapp.h.

7.6.3.20 typedef void(* objdestroy)(void *)

Definition at line 171 of file dtsapp.h.

7.6.3.21 typedef void(* radius_cb)(struct radius_packet *, void *)

Definition at line 133 of file dtsapp.h.

7.6.3.22 typedef struct radius_packet radius_packet

Forward decleration of structure.

Definition at line 122 of file dtsapp.h.

7.6.3.23 typedef void(* socketrecv)(struct fwsocket *, void *)

Definition at line 172 of file dtsapp.h.

7.6.3.24 typedef struct ssldata ssldata

Forward decleration of structure.

Definition at line 85 of file dtsapp.h.

7.6.3.25 typedef struct xml_doc xml_doc

Forward decleration of structure.

Definition at line 446 of file dtsapp.h.

7.6.3.26 typedef struct xml_node xml_node

Forward decleration of structure.

Definition at line 442 of file dtsapp.h.

7.6.3.27 typedef struct xml_search xml_search

Forward decleration of structure.

Definition at line 444 of file dtsapp.h.

7.6.3.28 typedef struct xslt_doc xslt_doc

Forward decleration of structure.

Definition at line 448 of file dtsapp.h.

7.6.4 Enumeration Type Documentation

7.6.4.1 enum ldap_attrtype

Enumerator:

LDAP_ATTRTYPE_CHAR

LDAP_ATTRTYPE_B64

LDAP_ATTRTYPE_OCTET

Definition at line 503 of file dtsapp.h.

```
                {
    LDAP_ATTRTYPE_CHAR,
    LDAP_ATTRTYPE_B64,
    LDAP_ATTRTYPE_OCTET
};
```

7.6.4.2 enum ldap_starttls

Enumerator:

LDAP_STARTTLS_NONE

LDAP_STARTTLS_ATTEMPT

LDAP_STARTTLS_ENFORCE

Definition at line 497 of file dtsapp.h.

```
                {
    LDAP_STARTTLS_NONE,
    LDAP_STARTTLS_ATTEMPT,
    LDAP_STARTTLS_ENFORCE
};
```

7.6.4.3 enum RADIUS_CODE

Enumerator:

RAD_CODE_AUTHREQUEST

RAD_CODE_AUTHACCEPT

RAD_CODE_AUTHREJECT

RAD_CODE_ACCTREQUEST

RAD_CODE_ACCTRESPONSE

RAD_CODE_AUTHCHALLENGE

Definition at line 391 of file dtsapp.h.

```
                {
    RAD_CODE_AUTHREQUEST     =   1,
    RAD_CODE_AUTHACCEPT  =   2,
    RAD_CODE_AUTHREJECT  =   3,
    RAD_CODE_ACCTREQUEST     =   4,
    RAD_CODE_ACCTRESPONSE  =   5,
    RAD_CODE_AUTHCHALLENGE     =   11
};
```

7.6.4.4 enum sock_flags

Enumerator:

SOCK_FLAG_BIND

SOCK_FLAG_CLOSE

Definition at line 87 of file dtsapp.h.

```
                {
    SOCK_FLAG_BIND         = 1 << 0,
    SOCK_FLAG_CLOSE        = 1 << 1
};
```

7.6.5 Function Documentation

7.6.5.1 void add_radserver (const char * *ipaddr,* const char * *auth,* const char * *acct,* const char * *secret,* int *timeout*)

Definition at line 233 of file radius.c.

References radius_server::acctport, addtobucket(), ALLOC_CONST, radius_server::authport, bucket_list_cnt(), create_bucketlist(), radius_server::id, radius_server::name, objalloc(), objunref(), radius_server::secret, radius_server::service, and radius_server::timeout.

```
                                                {
    struct radius_server *server;

    if ((server = objalloc(sizeof(*server), del_radserver))) {
        ALLOC_CONST(server->name, ipaddr);
        ALLOC_CONST(server->authport, auth);
        ALLOC_CONST(server->acctport, acct);
        ALLOC_CONST(server->secret, secret);
        if (!servers) {
            servers = create_bucketlist(0, hash_server);
        }
        server->id = bucket_list_cnt(servers);
        server->timeout = timeout;
        gettimeofday(&server->service, NULL);
        addtobucket(servers, server);
    }

    objunref(server);
}
```

7.6.5.2 unsigned char* addradattr (struct radius_packet * *packet,* char *type,* unsigned char * *val,* char *len*)

Definition at line 95 of file radius.c.

References radius_packet::attrs, radius_packet::len, and RAD_AUTH_HDR_LEN.

Referenced by addradattrint(), addradattrip(), and addradattrstr().

```
                                {
    unsigned char *data = packet->attrs + packet->len -
      RAD_AUTH_HDR_LEN;

    if (!len) {
        return NULL;
    }

    data[0] = type;
    data[1] = len + 2;
    if (val) {
        memcpy(data + 2, val, len);
    }

    packet->len += data[1];
    return (data);
}
```

7.6.5.3 void addradattrint (struct radius_packet * *packet,* char *type,* unsigned int *val*)

Definition at line 112 of file radius.c.

References addradattr().

```
        {
    unsigned int tval;

    tval = htonl(val);
    addradattr(packet, type, (unsigned char *)&tval, sizeof(tval));
}
```

7.6.5.4 void addradattrip (struct radius_packet * *packet,* char *type,* char * *ipaddr*)

Definition at line 119 of file radius.c.

References addradattr().

```
      {
    unsigned int tval;

    tval = inet_addr(ipaddr);
    addradattr(packet, type, (unsigned char *)&tval, sizeof(tval));
}
```

7.6.5.5 void addradattrstr (struct radius_packet * *packet,* char *type,* char * *str*)

Definition at line 126 of file radius.c.

References addradattr().

```
                                                                    {
    addradattr(packet, type, (unsigned char *)str, strlen(str));
}
```

7.6.5.6 void close_socket (struct fwsocket * *sock*)

Definition at line 57 of file socket.c.

References objunref(), setflag, and SOCK_FLAG_CLOSE.

```
                                                {
    if (sock) {
        setflag(sock, SOCK_FLAG_CLOSE);
        objunref(sock);
    }
}
```

7.6.5.7 struct fwsocket* dtls_listenssl (struct fwsocket * *sock*) [read]

Definition at line 530 of file sslutil.c.

References fwsocket::addr, ssldata::flags, make_socket(), objalloc(), objlock(), objunlock(), objunref(), fwsocket::proto, sockstruct::sa, fwsocket::sock, ssldata::ssl, fwsocket::ssl, SSL_DTLSCON, and fwsocket::type.

```
                                                    {
    struct ssldata *ssl = sock->ssl;
    struct ssldata *newssl;
    struct fwsocket *newsock;
    union sockstruct client;
```

```
#ifndef __WIN32__
    int on = 1;
#endif

    if (!(newssl = objalloc(sizeof(*newssl), free_ssldata))) {
        return NULL;
    }

    newssl->flags |= SSL_DTLSCON;

    dtlssetopts(newssl, ssl, sock);
    memset(&client, 0, sizeof(client));
    if (DTLSv1_listen(newssl->ssl, &client) <= 0) {
        objunref(newssl);
        return NULL;
    }

    objlock(sock);
    if (!(newsock = make_socket(sock->addr.sa.sa_family, sock
      ->type, sock->proto, newssl))) {
        objunlock(sock);
        objunref(newssl);
        return NULL;
    }
    objunlock(sock);
    memcpy(&newsock->addr, &client, sizeof(newsock->addr));
#ifndef __WIN32__
    setsockopt(newsock->sock, SOL_SOCKET, SO_REUSEADDR, &on, sizeof(on));
#ifdef SO_REUSEPORT
    setsockopt(newsock->sock, SOL_SOCKET, SO_REUSEPORT, &on, sizeof(on));
#endif
#endif
    objlock(sock);
    bind(newsock->sock, &sock->addr.sa, sizeof(sock->addr));
    objunlock(sock);
    connect(newsock->sock, &newsock->addr.sa, sizeof(newsock->addr
      ));

    dtlsaccept(newsock);

    return (newsock);
}
```

7.6.5.8 void* dtlsv1_init (const char * *cacert*, const char * *cert*, const char * *key*, int *verify*)

Definition at line 237 of file sslutil.c.

References ssldata::ctx, ssldata::ssl, and SSL_DTLSV1.

```
                {
    const SSL_METHOD *meth = DTLSv1_method();
    struct ssldata *ssl;

    ssl = sslinit(cacert, cert, key, verify, meth, SSL_DTLSV1);
    /* XXX BIO_CTRL_DGRAM_MTU_DISCOVER*/
    SSL_CTX_set_read_ahead(ssl->ctx, 1);

    return (ssl);
}
```

7.6.5.9 void ldap_close (struct ldap_conn * *ld*)

Definition at line 508 of file openldap.c.

References objunref().

```
                                        {
    objunref(ld);
}
```

7.6.5.10 struct ldap_conn* ldap_connect (const char * *uri,* enum ldap_starttls *starttls,* int *timelimit,* int *limit,* int *debug,* int * *err*) [read]

Definition at line 295 of file openldap.c.

References free_ldapconn(), ldap_conn::ldap, ldap_rebind_proc(), LDAP_STARTTLS_ENFORCE, LDAP_STARTTLS_NONE, ldap_conn::limit, objalloc(), objunref(), ldap_conn::sasl, ldap_conn::sctrlsp, ldap_conn::timelim, and ldap_conn::uri.

```
                                                                {
    struct ldap_conn *ld;
    int version = 3;
    int res, sslres;
    struct timeval timeout;

    if (!(ld = objalloc(sizeof(*ld), free_ldapconn))) {
        return NULL;
    }

    ld->uri = strdup(uri);
    ld->sctrlsp = NULL;
    ld->timelim = timelimit;
    ld->limit = limit;
    ld->sasl = NULL;

    if ((res = ldap_initialize(&ld->ldap, ld->uri) != LDAP_SUCCESS)) {
        objunref(ld);
        ld = NULL;
    } else {
        if (debug) {
            ldap_set_option(NULL, LDAP_OPT_DEBUG_LEVEL, &debug);
            ber_set_option(NULL, LBER_OPT_DEBUG_LEVEL, &debug);
        }
        if (timelimit) {
            timeout.tv_sec = timelimit;
            timeout.tv_usec = 0;
            ldap_set_option(ld->ldap, LDAP_OPT_NETWORK_TIMEOUT, (void *)&
      timeout);
        }
        ldap_set_option(ld->ldap, LDAP_OPT_PROTOCOL_VERSION, &version);
        ldap_set_option(ld->ldap, LDAP_OPT_REFERRALS, (void *)LDAP_OPT_ON);
        ldap_set_rebind_proc(ld->ldap, ldap_rebind_proc, ld
      );

        if ((starttls != LDAP_STARTTLS_NONE) & !
      ldap_tls_inplace(ld->ldap) && (sslres = ldap_start_tls_s(ld->ldap, ld->sctrlsp
      , NULL))) {
            if (starttls == LDAP_STARTTLS_ENFORCE) {
                objunref(ld);
                ld = NULL;
                res = sslres;
            }
        }
    }
    *err = res;
    return ld;
}
```

7.6.5.11 int ldap_domodify (struct ldap_conn * *ld,* struct ldap_modify * *lmod*)

Definition at line 1299 of file openldap.c.

References ldap_modify::bl, bucket_list_cnt(), ldap_modreq::cnt, ldap_modify::dn, init_bucket_loop(), ldap_conn::ldap, ldap_reqtoarr(), next_bucket_loop(), objlock(), objref(), objunlock(), objunref(), ldap_conn::sctrlsp, and stop_bucket_loop().

Referenced by ldap_mod_addattr(), ldap_mod_delattr(), and ldap_mod_repattr().

```
                                                                    {
    struct bucket_loop *bloop;
    struct ldap_modreq *modr;
    LDAPMod **modarr, **tmp, *item;
    int cnt, tot=0, res;

    if (!objref(ld)) {
        return LDAP_UNAVAILABLE;
```

```
    }

    for(cnt = 0; cnt < 3; cnt++) {
        tot += bucket_list_cnt(lmod->bl[cnt]);
    }
    tmp = modarr = calloc(sizeof(void *), (tot+1));

    for(cnt = 0; cnt < 3; cnt++) {
        bloop = init_bucket_loop(lmod->bl[cnt]);
        while(bloop && ((modr = next_bucket_loop(bloop)))) {
            if (!(item = ldap_reqtoarr(modr, cnt))) {
                ldap_mods_free(modarr, 1);
                objunref(ld);
                return LDAP_NO_MEMORY;
            }
            *tmp = item;
            tmp++;
            objunref(modr);
        }
        stop_bucket_loop(bloop);
    }
    *tmp = NULL;

    objlock(ld);
    res = ldap_modify_ext_s(ld->ldap, lmod->dn, modarr, ld->sctrlsp
      , NULL);
    objunlock(ld);
    ldap_mods_free(modarr, 1);
    objunref(ld);
    return res;
}
```

7.6.5.12 const char* ldap_errmsg (int *res*)

Definition at line 512 of file openldap.c.

```
                                        {
    return ldap_err2string(res);
}
```

7.6.5.13 struct ldap_attr* ldap_getattr (struct ldap_entry * *entry*, const char * *attr*) [read]

Definition at line 1090 of file openldap.c.

References ldap_entry::attrs, and bucket_list_find_key().

```
         {
    if (!entry || !entry->attrs) {
        return NULL;
    }
    return (struct ldap_attr *)bucket_list_find_key
      (entry->attrs, attr);
}
```

7.6.5.14 struct ldap_entry* ldap_getentry (struct ldap_results * *results*, const char * *dn*) [read]

Definition at line 1083 of file openldap.c.

References bucket_list_find_key(), and ldap_results::entries.

```
              {
    if (!results || !dn) {
        return NULL;
    }
    return (struct ldap_entry *)bucket_list_find_key
      (results->entries, dn);
}
```

7.6.5.15 int ldap_mod_add (struct ldap_modify * *lmod*, const char * *attr*, ...)

Definition at line 1207 of file openldap.c.

References add_modifyval(), getmodreq(), and objunref().

Referenced by ldap_mod_addattr().

```
                                                                        {
    va_list a_list;
    char *val;
    struct ldap_modreq *modr;

    if (!(modr = getmodreq(lmod, attr, LDAP_MOD_ADD))) {
        return 1;
    }

    va_start(a_list, attr);
    while((val = va_arg(a_list, void *))) {
        if (add_modifyval(modr, val)) {
            objunref(modr);
            return(1);
        }
    }

    objunref(modr);
    va_end(a_list);
    return 0;
}
```

7.6.5.16 int ldap_mod_addattr (struct ldap_conn * *ldap*, const char * *dn*, const char * *attr*, const char * *value*)

Definition at line 1359 of file openldap.c.

References ldap_domodify(), ldap_mod_add(), ldap_modifyinit(), and objunref().

```
                                {
    int res = 0;
    struct ldap_modify *lmod;

    if (!(lmod = ldap_modifyinit(dn))) {
        return LDAP_NO_MEMORY;
    }

    if (ldap_mod_add(lmod, attr, value, NULL)) {
        objunref(lmod);
        return LDAP_NO_MEMORY;
    }

    res = ldap_domodify(ldap, lmod);
    objunref(lmod);
    return res;
}
```

7.6.5.17 int ldap_mod_del (struct ldap_modify * *lmod*, const char * *attr*, ...)

Definition at line 1185 of file openldap.c.

References add_modifyval(), getmodreq(), and objunref().

Referenced by ldap_mod_delattr().

```
                                                                        {
    va_list a_list;
    char *val;
    struct ldap_modreq *modr;

    if (!(modr = getmodreq(lmod, attr, LDAP_MOD_DELETE))) {
        return 1;
    }

    va_start(a_list, attr);
    while((val = va_arg(a_list, void *))) {
```

```
        if (add_modifyval(modr, val)) {
            objunref(modr);
            return(1);
        }
    }

    objunref(modr);
    va_end(a_list);
    return 0;
}
```

7.6.5.18 int ldap_mod_delattr (struct ldap_conn * *ldap,* const char * *dn,* const char * *attr,* const char * *value*)

Definition at line 1338 of file openldap.c.

References ldap_domodify(), ldap_mod_del(), ldap_modifyinit(), and objunref().

Referenced by ldap_mod_remattr().

```
                                {
    struct ldap_modify *lmod;
    int res;

    if (!(lmod = ldap_modifyinit(dn))) {
        return LDAP_NO_MEMORY;
    }
    if (ldap_mod_del(lmod, attr, value, NULL)) {
        objunref(lmod);
        return LDAP_NO_MEMORY;
    }

    res = ldap_domodify(ldap, lmod);
    objunref(lmod);
    return res;
}
```

7.6.5.19 int ldap_mod_remattr (struct ldap_conn * *ldap,* const char * *dn,* const char * *attr*)

Definition at line 1355 of file openldap.c.

References ldap_mod_delattr().

```
            {
    return ldap_mod_delattr(ldap, dn, attr, NULL);
}
```

7.6.5.20 int ldap_mod_rep (struct ldap_modify * *lmod,* const char * *attr,* ...)

Definition at line 1229 of file openldap.c.

References add_modifyval(), getmodreq(), and objunref().

Referenced by ldap_mod_repattr().

```
                                                                    {
    va_list a_list;
    char *val;
    struct ldap_modreq *modr;

    if (!(modr = getmodreq(lmod, attr, LDAP_MOD_REPLACE))) {
        return 1;
    }

    va_start(a_list, attr);
    while((val = va_arg(a_list, void *))) {
        if (add_modifyval(modr, val)) {
            objunref(modr);
            return(1);
```

```
        }
    }

    objunref(modr);
    va_end(a_list);
    return 0;
}
```

7.6.5.21 int ldap_mod_repattr (struct ldap_conn * *ldap,* const char * *dn,* const char * *attr,* const char * *value*)

Definition at line 1377 of file openldap.c.

References ldap_domodify(), ldap_mod_rep(), ldap_modifyinit(), and objunref().

```
                                {
    struct ldap_modify *lmod;
    int res;

    if (!(lmod = ldap_modifyinit(dn))) {
        return LDAP_NO_MEMORY;
    }

    if (ldap_mod_rep(lmod, attr, value, NULL)) {
        objunref(lmod);
        return LDAP_NO_MEMORY;
    }

    res = ldap_domodify(ldap, lmod);
    objunref(lmod);
    return res;
}
```

7.6.5.22 struct ldap_modify* ldap_modifyinit (const char * *dn*) [read]

Definition at line 1097 of file openldap.c.

References ALLOC_CONST, ldap_modify::bl, create_bucketlist(), ldap_modify::dn, free_modify(), modify_hash(), objalloc(), and objunref().

Referenced by ldap_mod_addattr(), ldap_mod_delattr(), and ldap_mod_repattr().

```
                                                        {
    struct ldap_modify *mod;
    int cnt;

    if (!(mod = objalloc(sizeof(*mod), free_modify))) {
        return NULL;
    }

    ALLOC_CONST(mod->dn, dn);
    if (!mod->dn) {
        objunref(mod);
        return NULL;
    }

    for(cnt=0; cnt < 3; cnt++) {
        if (!(mod->bl[cnt] = create_bucketlist(4,
      modify_hash))) {
            objunref(mod);
            return NULL;
        }
    }

    return mod;
}
```

7.6.5.23 int ldap_saslbind (struct ldap_conn * *ld,* const char * *mech,* const char * *realm,* const char * *authcid,* const char * *passwd,* const char * *authzid*)

Definition at line 458 of file openldap.c.

References ALLOC_CONST, sasl_defaults::authcid, sasl_defaults::authzid, dts_sasl_interact(), free_sasl(), ldap_-conn::ldap, sasl_defaults::mech, objalloc(), objlock(), objref(), objunlock(), objunref(), sasl_defaults::passwd, sasl_-defaults::realm, ldap_conn::sasl, and ldap_conn::sctrlsp.

```
                                                                    {
    struct sasl_defaults *sasl;
    int res, sasl_flags = LDAP_SASL_AUTOMATIC | LDAP_SASL_QUIET;

    if (!objref(ld)) {
        return LDAP_UNAVAILABLE;
    }

    if (!(sasl = objalloc(sizeof(*sasl), free_sasl))) {
        return LDAP_NO_MEMORY;
    }

    ALLOC_CONST(sasl->passwd, passwd);

    if (mech) {
        ALLOC_CONST(sasl->mech, mech);
    } else {
        ldap_get_option(ld->ldap, LDAP_OPT_X_SASL_MECH, &sasl->mech);
    }

    if (realm) {
        ALLOC_CONST(sasl->realm, realm);
    } else {
        ldap_get_option(ld->ldap, LDAP_OPT_X_SASL_REALM, &sasl->realm
      );
    }

    if (authcid) {
        ALLOC_CONST(sasl->authcid, authcid);
    } else {
        ldap_get_option(ld->ldap, LDAP_OPT_X_SASL_AUTHCID, &sasl->authcid
     );
    }

    if (authzid) {
        ALLOC_CONST(sasl->authzid, authzid);
    } else {
        ldap_get_option(ld->ldap, LDAP_OPT_X_SASL_AUTHZID, &sasl->authzid
     );
    }

    objlock(ld);
    if (ld->sasl) {
        objunref(ld->sasl);
    }
    ld->sasl = sasl;
    res = ldap_sasl_interactive_bind_s(ld->ldap, NULL, sasl->mech, ld->
      sctrlsp , NULL, sasl_flags, dts_sasl_interact, sasl);
    objunlock(ld);
    objunref(ld);
    return res;
}
```

7.6.5.24 struct ldap_results* ldap_search_base (struct ldap_conn * *ld*, const char * *base*, const char * *filter*, int *b64enc*, int * *res*, ...) [read]

Definition at line 667 of file openldap.c.

References dts_ldapsearch(), and malloc.

```
                                                                    {
    va_list a_list;
    char *attr, **tmp, **attrs = NULL;
    int cnt = 1;

    va_start(a_list, res);
    while (( attr=va_arg(a_list, void *))) {
        cnt++;
    }
    va_end(a_list);

    if (cnt > 1) {
        tmp = attrs = malloc(sizeof(void *)*cnt);

        va_start(a_list, res);
```

```
        while (( attr=va_arg(a_list, char *))) {
            *tmp = attr;
            tmp++;
        }
        va_end(a_list);
        *tmp=NULL;
    }

    return dts_ldapsearch(ld, base, LDAP_SCOPE_BASE, filter,
      attrs, b64enc, res);
}
```

7.6.5.25 struct ldap_results* ldap_search_one (struct ldap_conn * *ld*, const char * *base*, const char * *filter*, int *b64enc*, int * *res*, ...) [read]

Definition at line 641 of file openldap.c.

References dts_ldapsearch(), and malloc.

```
{
    va_list a_list;
    char *attr, **tmp, **attrs = NULL;
    int cnt = 1;

    va_start(a_list, res);
    while (( attr=va_arg(a_list, void *))) {
        cnt++;
    }
    va_end(a_list);

    if (cnt > 1) {
        tmp = attrs = malloc(sizeof(void *)*cnt);

        va_start(a_list, res);
        while (( attr=va_arg(a_list, char *))) {
            *tmp = attr;
            tmp++;
        }
        va_end(a_list);
        *tmp=NULL;
    }

    return dts_ldapsearch(ld, base, LDAP_SCOPE_ONELEVEL, filter,
      attrs, b64enc, res);
}
```

7.6.5.26 struct ldap_results* ldap_search_sub (struct ldap_conn * *ld*, const char * *base*, const char * *filter*, int *b64enc*, int * *res*, ...) [read]

Definition at line 615 of file openldap.c.

References dts_ldapsearch(), and malloc.

Referenced by ldap_simplerebind().

```
{
    va_list a_list;
    char *attr, **tmp, **attrs = NULL;
    int cnt = 1;

    va_start(a_list, res);
    while (( attr=va_arg(a_list, void *))) {
        cnt++;
    }
    va_end(a_list);

    if (cnt > 1) {
        tmp = attrs = malloc(sizeof(void *)*cnt);

        va_start(a_list, res);
        while (( attr=va_arg(a_list, char *))) {
            *tmp = attr;
            tmp++;
```

```
        }
        va_end(a_list);
        *tmp=NULL;
    }

    return dts_ldapsearch(ld, base, LDAP_SCOPE_SUBTREE, filter,
      attrs, b64enc, res);
}
```

7.6.5.27 int ldap_simplebind (struct ldap_conn * *ld,* const char * *dn,* const char * *passwd*)

Definition at line 389 of file openldap.c.

References ldap_simple::cred, ldap_simple::dn, free_simple(), ldap_conn::ldap, malloc, objalloc(), objlock(), objref(), objunlock(), objunref(), ldap_conn::sctrlsp, and ldap_conn::simple.

Referenced by ldap_simplerebind().

```
        {
    struct ldap_simple *simple;
    struct berval *cred;
    int res, len = 0;

    if (!objref(ld)) {
        return LDAP_UNAVAILABLE;
    }

    if (passwd) {
        len = strlen(passwd);
    }
    simple = objalloc(sizeof(*simple), free_simple);
    cred = calloc(sizeof(*cred), 1);
    cred->bv_val = malloc(len);
    memcpy(cred->bv_val, passwd, len);
    cred->bv_len=len;
    simple->cred = cred;
    simple->dn = strdup(dn);

    objlock(ld);
    if (ld->simple) {
        objunref(ld->simple);
    }
    ld->simple = simple;
    res = ldap_sasl_bind_s(ld->ldap, simple->dn, LDAP_SASL_SIMPLE, simple
      ->cred, ld->sctrlsp, NULL, NULL);
    objunlock(ld);
    objunref(ld);
    return res;
}
```

7.6.5.28 int ldap_simplerebind (struct ldap_conn * *ld,* const char * *initialdn,* const char * *initialpw,* const char * *base,* const char * *filter,* const char * *uidrdn,* const char * *uid,* const char * *passwd*)

Definition at line 420 of file openldap.c.

References ldap_results::count, ldap_entry::dn, ldap_results::first_entry, ldap_search_sub(), ldap_simplebind(), malloc, objref(), and objunref().

```
        {
    int res, flen;
    struct ldap_results *results;
    const char *sfilt;

    if (!objref(ldap)) {
        return LDAP_UNAVAILABLE;
    }

    if ((res = ldap_simplebind(ldap, initialdn, initialpw))) {
        objunref(ldap);
        return res;
    }

    flen=strlen(uidrdn) + strlen(filter) + strlen(uid) + 7;
```

```
    sfilt = malloc(flen);
    snprintf((char *)sfilt, flen, "(&(%s=%s)%s)", uidrdn, uid, filter);

    if (!(results = ldap_search_sub(ldap, base, sfilt, 0, &res,
      uidrdn, NULL))) {
        free((void *)sfilt);
        objunref(ldap);
        return res;
    }
    free((void *)sfilt);

    if (results->count != 1) {
        objunref(results);
        objunref(ldap);
        return LDAP_INAPPROPRIATE_AUTH;
    }

    res = ldap_simplebind(ldap, results->first_entry
      ->dn, passwd);
    objunref(ldap);
    objunref(results);
    return res;
}
```

7.6.5.29 void ldap_unref_attr (struct ldap_entry * *entry*, struct ldap_attr * *attr*)

Definition at line 1053 of file openldap.c.

References ldap_entry::attrs, ldap_entry::first_attr, ldap_attr::next, objcnt(), objunref(), and remove_bucket_item().

```
                                                                {
    if (!entry || !attr) {
        return;
    }

    if (objcnt(attr) > 1) {
        objunref(attr);
    } else {
        if (attr == entry->first_attr) {
            entry->first_attr = attr->next;
        }
        remove_bucket_item(entry->attrs, attr);
    }
}
```

7.6.5.30 void ldap_unref_entry (struct ldap_results * *results*, struct ldap_entry * *entry*)

Definition at line 1068 of file openldap.c.

References ldap_results::entries, ldap_results::first_entry, ldap_entry::next, objcnt(), objunref(), and remove_bucket_item().

```
        {
    if (!results || !entry) {
        return;
    }

    if (objcnt(entry) > 1) {
        objunref(entry);
    } else {
        if (entry == results->first_entry) {
            results->first_entry = entry->next;
        }
        remove_bucket_item(results->entries, entry);
    }
}
```

7.6.5.31 struct fwsocket* make_socket (int *family*, int *type*, int *proto*, void * *ssl*) [read]

Definition at line 89 of file socket.c.

References objalloc(), objunref(), fwsocket::proto, fwsocket::sock, fwsocket::ssl, and fwsocket::type.

Referenced by dtls_listenssl().

```
    {
    struct fwsocket *si;

    if (!(si = objalloc(sizeof(*si),clean_fwsocket))) {
        return NULL;
    }

    if ((si->sock = socket(family, type, proto)) < 0) {
        objunref(si);
        return NULL;
    };

    if (ssl) {
        si->ssl = ssl;
    }
    si->type = type;
    si->proto = proto;

    return (si);
}
```

7.6.5.32 struct radius_packet* new_radpacket (unsigned char *code,* unsigned char *id*) [read]

Definition at line 171 of file radius.c.

References radius_packet::code, genrand(), radius_packet::len, malloc, RAD_AUTH_HDR_LEN, RAD_AUTH_TOKEN_LEN, and radius_packet::token.

```
    {
    struct radius_packet *packet;

    if ((packet = malloc(sizeof(*packet)))) {
        memset(packet, 0, sizeof(*packet));
        packet->len = RAD_AUTH_HDR_LEN;
        packet->code = code;
        genrand(&packet->token, RAD_AUTH_TOKEN_LEN
      );
    }
    return (packet);
}
```

7.6.5.33 struct nf_conntrack* nf_ctrack_buildct (uint8_t * *pkt*) [read]

Definition at line 90 of file nf_ctrack.c.

Referenced by nf_ctrack_delete(), and nf_ctrack_nat().

```
                                                        {
    struct nf_conntrack *ct;
    struct iphdr *ip = (struct iphdr *)pkt;
    union l4hdr *l4 = (union l4hdr *)(pkt + (ip->ihl * 4));

    if (!(ct = nfct_new())) {
        return (NULL);
    };

    /*Build tuple*/
    nfct_set_attr_u8(ct, ATTR_L3PROTO, PF_INET);
    nfct_set_attr_u32(ct, ATTR_IPV4_SRC, ip->saddr);
    nfct_set_attr_u32(ct, ATTR_IPV4_DST, ip->daddr);
    nfct_set_attr_u8(ct, ATTR_L4PROTO, ip->protocol);
    switch(ip->protocol) {
        case IPPROTO_TCP:
            nfct_set_attr_u16(ct, ATTR_PORT_SRC, l4->tcp.source);
            nfct_set_attr_u16(ct, ATTR_PORT_DST, l4->tcp.dest);
            break;
        case IPPROTO_UDP:
            nfct_set_attr_u16(ct, ATTR_PORT_SRC, l4->udp.source);
            nfct_set_attr_u16(ct, ATTR_PORT_DST, l4->udp.dest);
```

```
            break;
        case IPPROTO_ICMP:
            nfct_set_attr_u8(ct, ATTR_ICMP_TYPE, l4->icmp.type);
            nfct_set_attr_u8(ct, ATTR_ICMP_CODE, l4->icmp.code);
            nfct_set_attr_u16(ct, ATTR_ICMP_ID, l4->icmp.un.echo.id);
            /* no break */
        default
                :
            break;
    };

    return (ct);
}
```

7.6.5.34 void nf_ctrack_close (void)

Definition at line 275 of file nf_ctrack.c.

References ctrack, and objunref().

Referenced by nf_ctrack_delete(), nf_ctrack_dump(), and nf_ctrack_nat().

```
                                    {
    if (ctrack) {
        objunref(ctrack);
    }
    ctrack = NULL;
}
```

7.6.5.35 uint8_t nf_ctrack_delete (uint8_t * *pkt*)

Definition at line 126 of file nf_ctrack.c.

References ctrack, nf_ctrack_buildct(), nf_ctrack_close(), nf_ctrack_init(), nfct_struct::nfct, objlock(), and objunlock().

```
                                                {
    struct nf_conntrack *ct;
    uint8_t unref = 0;
    uint8_t ret = 0;

    if (!ctrack) {
        if (nf_ctrack_init()) {
            return (-1);
        }
        unref = 1;
    }

    ct = nf_ctrack_buildct(pkt);
    objlock(ctrack);
    if (nfct_query(ctrack->nfct, NFCT_Q_DESTROY, ct) < 0) {
        ret = -1;
    }
    objunlock(ctrack);
    nfct_destroy(ct);

    if (unref) {
        nf_ctrack_close();
    }

    return (ret);
}
```

7.6.5.36 void nf_ctrack_dump (void)

Definition at line 197 of file nf_ctrack.c.

References ctrack, nf_ctrack_close(), nf_ctrack_init(), nfct_struct::nfct, objlock(), and objunlock().

```
                                {
    uint32_t family = PF_INET;
    uint8_t unref = 0;

    if (!ctrack) {
        if (nf_ctrack_init()) {
            return;
        }
        unref = 1;
    }

    objlock(ctrack);
    nfct_callback_register(ctrack->nfct, NFCT_T_ALL, nfct_cb, NULL);
    nfct_query(ctrack->nfct, NFCT_Q_DUMP, &family);
    nfct_callback_unregister(ctrack->nfct);
    objunlock(ctrack);

    if (unref) {
        nf_ctrack_close();
    }
}
```

7.6.5.37 void nf_ctrack_endtrace (struct nfct_struct * *nfct*)

Definition at line 268 of file nf_ctrack.c.

References NFCTRACK_DONE, objunref(), and setflag.

```
                                                        {
    if (nfct) {
        setflag(nfct, NFCTRACK_DONE);
    }
    objunref(nfct);
}
```

7.6.5.38 uint8_t nf_ctrack_init (void)

Definition at line 83 of file nf_ctrack.c.

References ctrack.

Referenced by nf_ctrack_delete(), nf_ctrack_dump(), and nf_ctrack_nat().

```
                                    {
    if (!ctrack && !(ctrack = nf_ctrack_alloc(CONNTRACK, 0))) {
        return (-1);
    }
    return (0);
}
```

7.6.5.39 uint8_t nf_ctrack_nat (uint8_t * *pkt,* uint32_t *addr,* uint16_t *port,* uint8_t *dnat*)

Definition at line 153 of file nf_ctrack.c.

References ctrack, nf_ctrack_buildct(), nf_ctrack_close(), nf_ctrack_init(), nfct_struct::nfct, objlock(), and objunlock().

```
              {
    struct iphdr *ip = (struct iphdr *)pkt;
    struct nf_conntrack *ct;
    uint8_t unref = 0;
    uint8_t ret = 0;

    if (!ctrack) {
        if (nf_ctrack_init()) {
            return (-1);
        }
        unref = 1;
    }
```

```
    ct = nf_ctrack_buildct(pkt);
    nfct_setobjopt(ct, NFCT_SOPT_SETUP_REPLY);

    nfct_set_attr_u32(ct, ATTR_TIMEOUT, 120);
    nfct_set_attr_u32(ct, (dnat) ? ATTR_DNAT_IPV4 : ATTR_SNAT_IPV4, addr);

    switch(ip->protocol) {
        case IPPROTO_TCP:
            nfct_set_attr_u8(ct, ATTR_TCP_STATE, TCP_CONNTRACK_ESTABLISHED);
            /* no break */
        case IPPROTO_UDP:
            if (port) {
                nfct_set_attr_u16(ct, (dnat) ? ATTR_DNAT_PORT : ATTR_SNAT_PORT,
   port);
            }
            break;
    }

    objlock(ctrack);
    if (nfct_query(ctrack->nfct, NFCT_Q_CREATE_UPDATE, ct) < 0) {
        ret = -1;
    }
    objunlock(ctrack);
    nfct_destroy(ct);

    if (unref) {
        nf_ctrack_close();
    }

    return (ret);
}
```

7.6.5.40 struct nfct_struct* nf_ctrack_trace (void) [read]

Definition at line 254 of file nf_ctrack.c.

References framework_mkthread(), nfct_struct::nfct, and objunref().

```
                                        {
    struct nfct_struct *nfct;

    if (!(nfct = nf_ctrack_alloc(CONNTRACK, NFCT_ALL_CT_GROUPS))) {
        return (NULL);
    }

    if (!framework_mkthread(nf_ctrack_trace_th, NULL, NULL,
      nfct)) {
        objunref(nfct);
        return (NULL);
    }
    return (nfct);
}
```

7.6.5.41 struct nfq_queue* nfqueue_attach (uint16_t *pf*, uint16_t *num*, uint8_t *mode*, uint32_t *range*, nfqueue_cb *cb*, void * *data*) [read]

Definition at line 225 of file nf_queue.c.

References bucket_list_find_key(), nfq_queue::cb, nfq_queue::data, nfq_struct::h, nfq_queue::nfq, nfqueues, objalloc(), objlock(), objunlock(), objunref(), nfq_queue::qh, and nfq_list::queues.

```
                                                {
    struct nfq_queue *nfq_q;

    if (!(nfq_q = objalloc(sizeof(*nfq_q), nfqueue_close_q))) {
        return (NULL);
    }

    objlock(nfqueues);
    if (!(nfqueues && (nfq_q->nfq = bucket_list_find_key
      (nfqueues->queues, &pf))) &&
            !(nfq_q->nfq || (nfq_q->nfq = nfqueue_init(pf)))) {
        objunlock(nfqueues);
```

```
        objunref(nfq_q);
        return (NULL);
    }
    objunlock(nfqueues);

    if (!(nfq_q->qh = nfq_create_queue(nfq_q->nfq->h, num, &
      nfqueue_callback, nfq_q))) {
        objunref(nfq_q);
        return (NULL);
    }

    if (cb) {
        nfq_q->cb = cb;
    }

    if (data) {
        nfq_q->data = data;
    }

    nfq_set_mode(nfq_q->qh, mode, range);

    return (nfq_q);
}
```

7.6.5.42 unsigned char* radius_attr_first (struct radius_packet * *packet*)

Definition at line 560 of file radius.c.

References radius_packet::attrs.

```
                                                                    {
    return (packet->attrs);
}
```

7.6.5.43 unsigned char* radius_attr_next (struct radius_packet * *packet,* unsigned char * *attr*)

Definition at line 564 of file radius.c.

References radius_packet::attrs, radius_packet::len, and RAD_AUTH_HDR_LEN.

```
                {
    int offset = (packet->len - RAD_AUTH_HDR_LEN) - (attr -
      packet->attrs);

    if (!(offset - attr[1])) {
        return NULL;
    }

    return (attr + attr[1]);
}
```

7.6.5.44 void rfc6296_map (struct natmap * *map,* struct in6_addr * *ipaddr,* int *out*)

Definition at line 47 of file rfc6296.c.

References natmap::adji, natmap::adjo, natmap::epre, natmap::ipre, and natmap::mask.

```
                                                                    {
    uint16_t *addr_16 = (uint16_t *)&ipaddr->s6_addr;
    uint32_t calc;
    uint8_t cnt, *prefix, bitlen, bytelen;
    uint16_t adj;

    prefix = (out) ? map->epre : map->ipre;
    adj = (out) ? map->adjo : map->adji;

    if ((bitlen = map->mask % 8)) {
        bytelen = (map->mask - bitlen) / 8;
        bytelen++;
    } else {
```

```
        bytelen = map->mask / 8;
    }

    /*as per RFC we handle /48 and longer /48 changes are reflected in SN*/
    if ((bytelen == 6) && (~addr_16[3]) && (!bitlen)) {
        memcpy(&ipaddr->s6_addr, prefix, bytelen);
        calc = ntohs(addr_16[3]) + adj;
        addr_16[3] = htons((calc & 0xFFFF) + (calc >> 16));
        if (! ~addr_16[3]) {
            addr_16[3] = 0;
        }
    } else
        if ((bytelen > 6) && (bytelen < 15)) {
            /* find first non 0xFFFF word in lower 64 bits*/
            for(cnt = ((bytelen-1) >> 1) + 1; cnt < 8; cnt++) {
                if (! ~addr_16[cnt]) {
                    continue;
                }
                if (bitlen) {
                    ipaddr->s6_addr[bytelen-1] = prefix[bytelen-1] | (ipaddr->
      s6_addr[bytelen-1] & ((1 << (8 - bitlen)) -1));
                } else {
                    ipaddr->s6_addr[bytelen-1] = prefix[bytelen-1];
                }
                memcpy(&ipaddr->s6_addr, prefix, bytelen - 1);
                calc = ntohs(addr_16[cnt]) + adj;
                addr_16[cnt] = htons((calc & 0xFFFF) + (calc >> 16));
                if (! ~addr_16[cnt]) {
                    addr_16[cnt] = 0;
                }
                break;
            }
        }
}
```

7.6.5.45 int rfc6296_map_add (char * *intaddr*, char * *extaddr*)

Definition at line 94 of file rfc6296.c.

References addtobucket(), natmap::adji, natmap::adjo, checksum(), create_bucketlist(), natmap::epre, natmap::ipre, natmap::mask, objalloc(), and objunref().

```
                                                        {
    struct natmap *map;
    uint16_t emask, imask, isum, esum, bytelen, bitlen;
    char inip[43], exip[43], *tmp2;
    struct in6_addr i6addr;
    uint32_t adj;

    strncpy(inip, intaddr, 43);
    if ((tmp2 = rindex(inip, '/'))) {
        tmp2[0] = '\0';
        tmp2++;
        imask = atoi(tmp2);
    } else {
        return (-1);
    }

    strncpy(exip, extaddr, 43);
    if ((tmp2 = rindex(exip, '/'))) {
        tmp2[0] = '\0';
        tmp2++;
        emask = atoi(tmp2);
    } else {
        return (-1);
    }

    map = objalloc(sizeof(*map), NULL);
    map->mask = (emask > imask) ? emask : imask;

    /*rfc says we must zero extend this is what we do here looking at each
       supplied len*/
    /*external range*/
    inet_pton(AF_INET6, exip, &i6addr);
    if ((bitlen = emask % 8)) {
        bytelen = (emask - bitlen) / 8;
        i6addr.s6_addr[bytelen] &= ~((1 << (8 - bitlen)) - 1);
        bytelen++;
    } else {
        bytelen = emask / 8;
    }
```

```
    memcpy(map->epre, &i6addr.s6_addr, bytelen);

    /*internal range*/
    inet_pton(AF_INET6, inip, &i6addr);
    if ((bitlen = imask % 8)) {
        bytelen = (imask - bitlen) / 8;
        i6addr.s6_addr[bytelen] &= ~((1 << (8 - bitlen)) - 1);
        bytelen++;
    } else {
        bytelen = imask / 8;
    }
    memcpy(map->ipre, &i6addr.s6_addr, bytelen);

    /*calculate the adjustments from checksums of prefixes*/
    if ((bitlen = map->mask % 8)) {
        bytelen = (map->mask - bitlen) / 8;
        bytelen++;
    } else {
        bytelen = map->mask / 8;
    }
    esum = ntohs(checksum(map->epre, bytelen));
    isum = ntohs(checksum(map->ipre, bytelen));

    /*outgoing transform*/
    adj = esum - isum;
    adj = (adj & 0xFFFF) + (adj >> 16);
    map->adjo = (uint16_t)adj;

    /*incoming transform*/
    adj = isum - esum;
    adj = (adj & 0xFFFF) + (adj >> 16);
    map->adji = (uint16_t)adj;

    if (!nptv6tbl && (!(nptv6tbl = create_bucketlist
      (5, nptv6_hash)))) {
        objunref(map);
        return (-1);
    }
    addtobucket(nptv6tbl, map);
    objunref(map);

    return (0);
}
```

7.6.5.46 int send_radpacket (struct radius_packet * *packet*, const char * *userpass*, radius_cb *read_cb*, void * *cb_data*)

Definition at line 390 of file radius.c.

```
                                        {
    return (_send_radpacket(packet, userpass, NULL, read_cb, cb_data));
}
```

7.6.5.47 uint16_t snprintf_pkt (struct nfq_data * *tb*, struct nfqnl_msg_packet_hdr * *ph*, uint8_t * *pkt*, char * *buff*, uint16_t *len*)

Definition at line 259 of file nf_queue.c.

References id.

```
                                                    {
    struct iphdr *ip = (struct iphdr *)pkt;
    char *tmp = buff;
    uint32_t id, mark, ifi;
    uint16_t tlen, left = len;
    char saddr[INET_ADDRSTRLEN], daddr[INET_ADDRSTRLEN];

    if (ph) {
        id = ntohl(ph->packet_id);
        snprintf(tmp, left, "hw_protocol=0x%04x hook=%u id=%u ",
                 ntohs(ph->hw_protocol), ph->hook, id);
        tlen = strlen(tmp);
        tmp += tlen;
        left -= tlen;
    }
```

```
    if ((mark = nfq_get_nfmark(tb))) {
        snprintf(tmp, left, "mark=%u ", mark);
        tlen = strlen(tmp);
        tmp += tlen;
        left -= tlen;
    }

    if ((ifi = nfq_get_indev(tb))) {
        snprintf(tmp, left, "indev=%u ", ifi);
        tlen = strlen(tmp);
        tmp += tlen;
        left -= tlen;
    }

    if ((ifi = nfq_get_outdev(tb))) {
        snprintf(tmp, left, "outdev=%u ", ifi);
        tlen = strlen(tmp);
        tmp += tlen;
        left -= tlen;
    }

    if (pkt && (ip->version == 4)) {
        union l4hdr *l4 = (union l4hdr *)(pkt + (ip->ihl*4));

        inet_ntop(AF_INET, &ip->saddr, saddr, INET_ADDRSTRLEN);
        inet_ntop(AF_INET, &ip->daddr, daddr, INET_ADDRSTRLEN);

        snprintf(tmp, left, "src=%s dst=%s proto=%i ", saddr, daddr, ip->
      protocol);
        tlen = strlen(tmp);
        tmp += tlen;
        left -= tlen;

        switch(ip->protocol) {
            case IPPROTO_TCP:
                snprintf(tmp, left, "sport=%i dport=%i ", ntohs(l4->tcp.source)
      , ntohs(l4->tcp.dest));
                break;
            case IPPROTO_UDP:
                snprintf(tmp, left, "sport=%i dport=%i ", ntohs(l4->udp.source)
      , ntohs(l4->udp.dest));
                break;
            case IPPROTO_ICMP:
                snprintf(tmp, left, "type=%i code=%i id=%i ", l4->icmp.type, l4
      ->icmp.code, ntohs(l4->icmp.un.echo.id));
                break;
        }
        tlen = strlen(tmp);
        tmp += tlen;
        left -= tlen;
    }

    return (len - left);
}
```

7.6.5.48 struct fwsocket* sockbind (int *family,* int *stype,* int *proto,* const char * *ipaddr,* const char * *port,* void * *ssl,* int *backlog*) [read]

Definition at line 212 of file socket.c.

```
                                                                {
    return(_opensocket(family, stype, proto, ipaddr, port, ssl, 1, backlog));
}
```

7.6.5.49 struct fwsocket* sockconnect (int *family,* int *stype,* int *proto,* const char * *ipaddr,* const char * *port,* void * *ssl*) [read]

Definition at line 200 of file socket.c.

```
                                                    {
    return(_opensocket(family, stype, proto, ipaddr, port, ssl, 0, 0));
}
```

7.6.5.50 void socketclient (struct fwsocket * *sock,* void * *data,* socketrecv *read,* threadcleanup *cleanup*)

Definition at line 374 of file socket.c.

References startsslclient().

```
                          {
    startsslclient(sock);

    _start_socket_handler(sock, read, NULL, cleanup, data);
}
```

7.6.5.51 int socketread (struct fwsocket * *sock,* void * *buf,* int *num*)

Definition at line 362 of file sslutil.c.

References socketread_d().

```
                                                                {
    return (socketread_d(sock, buf, num, NULL));
}
```

7.6.5.52 int socketread_d (struct fwsocket * *sock,* void * *buf,* int *num,* union sockstruct * *addr*)

Definition at line 297 of file sslutil.c.

References fwsocket::flags, objlock(), objunlock(), objunref(), sockstruct::sa, fwsocket::sock, SOCK_FLAG_CLOSE, ssldata::ssl, fwsocket::ssl, and fwsocket::type.

Referenced by socketread().

```
              {
    struct ssldata *ssl = sock->ssl;
    socklen_t salen = sizeof(*addr);
    int ret, err, syserr;

    if (!ssl || !ssl->ssl) {
        objlock(sock);
        if (addr && (sock->type == SOCK_DGRAM)) {
            ret = recvfrom(sock->sock, buf, num, 0, &addr->sa, &salen);
        } else {
            ret = read(sock->sock, buf, num);
        }
        if (ret == 0) {
            sock->flags |= SOCK_FLAG_CLOSE;
        }
        objunlock(sock);
        return (ret);
    }

    objlock(ssl);
    /* ive been shutdown*/
    if (!ssl->ssl) {
        objunlock(ssl);
        return (-1);
    }
    ret = SSL_read(ssl->ssl, buf, num);
    err = SSL_get_error(ssl->ssl, ret);
    if (ret == 0) {
        sock->flags |= SOCK_FLAG_CLOSE;
    }
    objunlock(ssl);
    switch (err) {
        case SSL_ERROR_NONE:
            break;
        case SSL_ERROR_WANT_X509_LOOKUP:
            printf("Want X509\n");
            break;
        case SSL_ERROR_WANT_READ:
            printf("Want Read\n");
            break;
```

```
        case SSL_ERROR_WANT_WRITE:
            printf("Want write\n");
            break;
        case SSL_ERROR_ZERO_RETURN:
        case SSL_ERROR_SSL:
            objlock(sock);
            objunref(sock->ssl);
            sock->ssl = NULL;
            objunlock(sock);
            break;
        case SSL_ERROR_SYSCALL:
            syserr = ERR_get_error();
            if (syserr || (!syserr && (ret == -1))) {
                printf("R syscall %i %i\n", syserr, ret);
            }
            break;
        default
                :
            printf("other\n");
            break;
    }

    return (ret);
}
```

7.6.5.53 void socketserver (struct fwsocket * *sock*, socketrecv *connectfunc*, socketrecv *acceptfunc*, threadcleanup *cleanup*, void * *data*)

Definition at line 354 of file socket.c.

References fwsocket::children, create_bucketlist(), dtsl_serveropts(), fwsocket::flags, objlock(), objunlock(), fwsocket::ssl, and fwsocket::type.

```
          {

    objlock(sock);
    if (sock->flags & SOCK_FLAG_BIND) {
        if (sock->ssl || !(sock->type == SOCK_DGRAM)) {
            sock->children = create_bucketlist(6,
      hash_socket);
        }
        if (sock->ssl && (sock->type == SOCK_DGRAM)) {
            objunlock(sock);
            dtsl_serveropts(sock);
        } else {
            objunlock(sock);
        }
    } else {
        objunlock(sock);
    }
    _start_socket_handler(sock, read, acceptfunc, cleanup, data);
}
```

7.6.5.54 int socketwrite (struct fwsocket * *sock*, const void * *buf*, int *num*)

Definition at line 444 of file sslutil.c.

References socketwrite_d().

```
                                                                    {
    return (socketwrite_d(sock, buf, num, NULL));
}
```

7.6.5.55 int socketwrite_d (struct fwsocket * *sock*, const void * *buf*, int *num*, union sockstruct * *addr*)

Definition at line 366 of file sslutil.c.

References fwsocket::flags, objlock(), objunlock(), objunref(), sockstruct::sa, setflag, fwsocket::sock, SOCK_FLAG_CLOSE, ssldata::ssl, fwsocket::ssl, and fwsocket::type.

Referenced by socketwrite().

```
                        {
    struct ssldata *ssl = (sock) ? sock->ssl : NULL;
    int ret, err, syserr;

    if (!sock) {
        return (-1);
    }

#ifndef __WIN32__
    if (!ssl || !ssl->ssl) {
        objlock(sock);
        if (addr && (sock->type == SOCK_DGRAM)) {
            ret = sendto(sock->sock, buf, num, MSG_NOSIGNAL, &addr->sa,
      sizeof(*addr));
        } else {
            ret = send(sock->sock, buf, num, MSG_NOSIGNAL);
        }
        if (ret == -1) {
            switch(errno) {
                case EBADF:
                case EPIPE:
                case ENOTCONN:
                case ENOTSOCK:
                    sock->flags |= SOCK_FLAG_CLOSE;
                    break;
            }
        }
        objunlock(sock);
        return (ret);
    }
#endif

    objlock(ssl);
    if (SSL_state(ssl->ssl) != SSL_ST_OK) {
        objunlock(ssl);
        return (SSL_ERROR_SSL);
    }
    ret = SSL_write(ssl->ssl, buf, num);
    err = SSL_get_error(ssl->ssl, ret);
    objunlock(ssl);

    if (ret == -1) {
        setflag(sock, SOCK_FLAG_CLOSE);
    }

    switch(err) {
        case SSL_ERROR_NONE:
            break;
        case SSL_ERROR_WANT_READ:
            printf("Want Read\n");
            break;
        case SSL_ERROR_WANT_WRITE:
            printf("Want write\n");
            break;
        case SSL_ERROR_WANT_X509_LOOKUP:
            printf("Want X509\n");
            break;
        case SSL_ERROR_ZERO_RETURN:
        case SSL_ERROR_SSL:
            objlock(sock);
            objunref(sock->ssl);
            sock->ssl = NULL;
            objunlock(sock);
            break;
        case SSL_ERROR_SYSCALL:
            syserr = ERR_get_error();
            if (syserr || (!syserr && (ret == -1))) {
                printf("W syscall %i %i\n", syserr, ret);
            }
            break;
        default
                :
            printf("other\n");
            break;
    }

    return (ret);
}
```

7.6.5.56 void ssl_shutdown (void * *ssl*)

Definition at line 92 of file sslutil.c.

References objlock(), objunlock(), and ssldata::ssl.

```
                                        {
    struct ssldata *ssl = data;
    int err, ret;

    if (!ssl) {
        return;
    }

    objlock(ssl);
    if (ssl->ssl && ((ret = SSL_shutdown(ssl->ssl)) < 1)) {
        objunlock(ssl);
        if (ret == 0) {
            objlock(ssl);
            ret = SSL_shutdown(ssl->ssl);
        } else {
            objlock(ssl);
        }
        err = SSL_get_error(ssl->ssl, ret);
        switch(err) {
            case SSL_ERROR_WANT_READ:
                printf("SSL_shutdown wants read\n");
                break;
            case SSL_ERROR_WANT_WRITE:
                printf("SSL_shutdown wants write\n");
                break;
            case SSL_ERROR_SSL:
                /*ignore im going away now*/
            case SSL_ERROR_SYSCALL:
                /* ignore this as documented*/
            case SSL_ERROR_NONE:
                /* nothing to see here moving on*/
                break;
            default
                    :
                printf("SSL Shutdown unknown error %i\n", err);
                break;
        }
    }
    if (ssl->ssl) {
        SSL_free(ssl->ssl);
        ssl->ssl = NULL;
    }
    objunlock(ssl);
}
```

7.6.5.57 void sslstartup (void)

Definition at line 448 of file sslutil.c.

References COOKIE_SECRET_LENGTH, genrand(), and malloc.

Referenced by framework_init().

```
                                {
    SSL_library_init();
    SSL_load_error_strings();
    OpenSSL_add_ssl_algorithms();

    if ((cookie_secret = malloc(COOKIE_SECRET_LENGTH)
      )) {
        genrand(cookie_secret, COOKIE_SECRET_LENGTH)
      ;
    }
}
```

7.6.5.58 void* sslv2_init (const char * *cacert*, const char * *cert*, const char * *key*, int *verify*)

Definition at line 221 of file sslutil.c.

References SSL_SSLV2.

```
                    {
```

```
    const SSL_METHOD *meth = SSLv2_method();

    return (sslinit(cacert, cert, key, verify, meth, SSL_SSLV2));
}
```

7.6.5.59 void* sslv3_init (const char * *cacert*, const char * *cert*, const char * *key*, int *verify*)

Definition at line 228 of file sslutil.c.

References ssldata::ssl, and SSL_SSLV3.

```
                {
    const SSL_METHOD *meth = SSLv3_method();
    struct ssldata *ssl;

    ssl = sslinit(cacert, cert, key, verify, meth, SSL_SSLV3);

    return (ssl);
}
```

7.6.5.60 void startsslclient (struct fwsocket * *sock*)

Definition at line 602 of file sslutil.c.

References ssldata::flags, fwsocket::ssl, SSL_SERVER, and fwsocket::type.

Referenced by socketclient().

```
                                                {
    if (!sock || !sock->ssl || (sock->ssl->flags & SSL_SERVER
      )) {
        return;
    }

    switch(sock->type) {
        case SOCK_DGRAM:
            dtlsconnect(sock);
            break;
        case SOCK_STREAM:
            sslsockstart(sock, NULL, 0);
            break;
    }
}
```

7.6.5.61 struct fwsocket* tcpbind (const char * *ipaddr*, const char * *port*, void * *ssl*, int *backlog*) [read]

Definition at line 220 of file socket.c.

```
                    {
    return (_opensocket(PF_UNSPEC, SOCK_STREAM, IPPROTO_TCP, ipaddr, port, ssl
      , 1, backlog));
}
```

7.6.5.62 struct fwsocket* tcpconnect (const char * *ipaddr*, const char * *port*, void * *ssl*) [read]

Definition at line 208 of file socket.c.

```
           {
    return (_opensocket(PF_UNSPEC, SOCK_STREAM, IPPROTO_TCP, ipaddr, port, ssl
      , 0, 0));
}
```

7.6.5.63 void tlsaccept (struct fwsocket * *sock,* struct ssldata * *orig*)

Definition at line 290 of file sslutil.c.

References objalloc(), and fwsocket::ssl.

```
                                                                    {
    if ((sock->ssl = objalloc(sizeof(*sock->ssl), free_ssldata)))
       {
        sslsockstart(sock, orig, 1);
    }

}
```

7.6.5.64 void* tlsv1_init (const char * *cacert,* const char * *cert,* const char * *key,* int *verify*)

Definition at line 214 of file sslutil.c.

References SSL_TLSV1.

```
                {
    const SSL_METHOD *meth = TLSv1_method();

    return (sslinit(cacert, cert, key, verify, meth, SSL_TLSV1));
}
```

7.6.5.65 struct fwsocket* udpbind (const char * *ipaddr,* const char * *port,* void * *ssl*) [read]

Definition at line 216 of file socket.c.

```
       {
    return (_opensocket(PF_UNSPEC, SOCK_DGRAM, IPPROTO_UDP, ipaddr, port, ssl
      , 1, 0));
}
```

7.6.5.66 struct fwsocket* udpconnect (const char * *ipaddr,* const char * *port,* void * *ssl*) [read]

Definition at line 204 of file socket.c.

Referenced by radconnect().

```
          {
    return (_opensocket(PF_UNSPEC, SOCK_DGRAM, IPPROTO_UDP, ipaddr, port, ssl
      , 0, 0));
}
```

7.7 src/include/list.h File Reference

Data Structures

- struct linkedlist
- struct hashedlist

Macros

- #define LIST_INIT(head, entry)
- #define LIST_ADD(head, entry)
- #define LIST_ADD_HASH(head, entry, bhash)
- #define LIST_FORLOOP(head, entry, cur) for(cur = head; (cur && (entry = cur->data)); cur = cur->next)
- #define LIST_FORLOOP_SAFE(head, entry, cur, tmp) for(cur = head; (cur && (entry = cur->data) && ((tmp = cur->next) || 1)); cur = tmp)
- #define LIST_REMOVE_ENTRY(head, cur)
- #define LIST_REMOVE_ITEM(head, entry)
- #define LIST_FOREACH_START_SAFE(head, entry)
- #define LIST_FOREACH_START(head, entry)
- #define LIST_FOREACH_END }
- #define LIST_REMOVE_CURRENT(head) LIST_REMOVE_ENTRY(head, _cur_head)

7.7.1 Macro Definition Documentation

7.7.1.1 #define LIST_ADD(*head, entry*)

Value:

```
{ \
    __typeof(head) _tmp_head = head; \
    if (!_tmp_head) { \
        LIST_INIT(_tmp_head, entry) \
        head = _tmp_head; \
    } else if (_tmp_head->prev) {\
        __typeof(head) _ent_head = NULL; \
        LIST_INIT(_ent_head, entry) \
        if (_ent_head) { \
            _ent_head->prev = _tmp_head->prev; \
            _tmp_head->prev->next = _ent_head; \
            _tmp_head->prev = _ent_head; \
        } \
    } else { \
        _tmp_head->data = entry; \
        _tmp_head->next = NULL; \
        _tmp_head->prev = _tmp_head; \
    } \
}
```

Definition at line 62 of file list.h.

7.7.1.2 #define LIST_ADD_HASH(*head, entry, bhash*)

Value:

```
{ \
    LIST_ADD(head, entry); \
    if (head->prev->data == entry) { \
        head->hash = bhash; \
    } \
}
```

Definition at line 82 of file list.h.

7.7.1.3 #define LIST_FOREACH_END }

Definition at line 147 of file list.h.

7.7.1.4 #define LIST_FOREACH_START(*head, entry*)

Value:

```
{ \
    __typeof(head) _cur_head; \
    LIST_FORLOOP(head, entry, _cur_head)
```

Definition at line 143 of file list.h.

7.7.1.5 #define LIST_FOREACH_START_SAFE(*head, entry*)

Value:

```
{ \
    __typeof(head) _tmp_head, _cur_head; \
    LIST_FORLOOP_SAFE(head, entry, _cur_head, _tmp_head)
```

Definition at line 139 of file list.h.

7.7.1.6 #define LIST_FORLOOP(*head, entry, cur*) for(cur = head; (cur && (entry = cur->data)); cur = cur->next)

Definition at line 93 of file list.h.

7.7.1.7 #define LIST_FORLOOP_SAFE(*head, entry, cur, tmp*) for(cur = head; (cur && (entry = cur->data) && ((tmp = cur->next) || 1)); cur = tmp)

Definition at line 96 of file list.h.

7.7.1.8 #define LIST_INIT(*head, entry*)

Value:

```
{ \
    if (!head && (!(head = malloc(sizeof(*head))))) { \
        printf("Could not allocate memory for list head\n"); \
    } else { \
        head->data = entry; \
        head->next = NULL; \
        head->prev = (entry) ? head : NULL; \
    } \
}
```

Definition at line 48 of file list.h.

7.7.1.9 #define LIST_REMOVE_CURRENT(*head*) LIST_REMOVE_ENTRY(head, _cur_head)

Definition at line 149 of file list.h.

7.7.1.10 #define LIST_REMOVE_ENTRY(*head, cur*)

Value:

```
{ \
    if (cur && (head == cur)) { \
        cur->prev->next = NULL; \
        if (cur->next) { \
            cur->next->prev = cur->prev; \
        } \
```

```
        head = cur->next; \
        free(cur); \
    } else if (cur->next) { \
        cur->prev->next = cur->next; \
        cur->next->prev = cur->prev; \
        free(cur); \
    } else { \
        cur->prev->next = NULL; \
        head->prev = cur->prev; \
        free(cur); \
    } \
}
```

Definition at line 99 of file list.h.

7.7.1.11 #define LIST_REMOVE_ITEM(*head, entry*)

Value:

```
{\
    __typeof(head) _tmp_head, _cur_head; \
    __typeof(entry) _tmp_ent; \
    LIST_FORLOOP_SAFE(head, _tmp_ent, _cur_head, _tmp_head) {
      \
        if (entry == _tmp_ent) { \
            LIST_REMOVE_ENTRY(head, _cur_head); \
            break; \
        } \
    } \
}
```

Definition at line 123 of file list.h.

7.8 src/include/priv_xml.h File Reference

Data Structures

- struct xml_buffer
- struct xml_doc

Functions

- void free_buffer (void *data)

7.9 src/include/private.h File Reference

Functions

- void dtsl_serveropts (struct fwsocket *sock)
- void dtlshandltimeout (struct fwsocket *sock)
- int startthreads (void)

  initialise the threadlist start manager thread
- void jointhreads (void)

  Join the manager thread.
- int thread_signal (int sig)

7.9.1 Function Documentation

7.9.1.1 void dtlshandltimeout (struct fwsocket * *sock*)

Definition at line 630 of file sslutil.c.

References objlock(), objunlock(), ssldata::ssl, and fwsocket::ssl.

```
                                                    {
    if (!sock->ssl) {
        return;
    }

    objlock(sock->ssl);
    DTLSv1_handle_timeout(sock->ssl->ssl);
    objunlock(sock->ssl);
}
```

7.9.1.2 void dtsl_serveropts (struct fwsocket * *sock*)

Definition at line 489 of file sslutil.c.

References ssldata::ctx, ssldata::flags, objlock(), objunlock(), ssldata::ssl, fwsocket::ssl, and SSL_SERVER.

Referenced by socketserver().

```
                                                    {
    struct ssldata *ssl = sock->ssl;

    if (!ssl) {
        return;
    }

    dtlssetopts(ssl, NULL, sock);

    objlock(ssl);
    SSL_CTX_set_cookie_generate_cb(ssl->ctx, generate_cookie);
    SSL_CTX_set_cookie_verify_cb(ssl->ctx, verify_cookie);
    SSL_CTX_set_session_cache_mode(ssl->ctx, SSL_SESS_CACHE_OFF);

    SSL_set_options(ssl->ssl, SSL_OP_COOKIE_EXCHANGE);
    ssl->flags |= SSL_SERVER;
    objunlock(ssl);
}
```

7.9.1.3 int thread_signal (int *sig*)

7.10 src/interface.c File Reference

Wrapper arround Linux libnetlink for managing network interfaces.

```
#include <netinet/in.h>
#include <linux/if_vlan.h>
#include <linux/if_ether.h>
#include <linux/if_packet.h>
#include <linux/if_tun.h>
#include <linux/if_arp.h>
#include <linux/sockios.h>
#include <linux/if.h>
#include <sys/ioctl.h>
#include <sys/time.h>
#include <fcntl.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include "include/dtsapp.h"
#include "libnetlink/include/libnetlink.h"
#include "libnetlink/include/ll_map.h"
#include "libnetlink/include/utils.h"
```

Data Structures

- struct iplink_req
- struct ipaddr_req

Functions

- void closenetlink ()
- int get_iface_index (const char *ifname)
- int delete_kernvlan (char *ifname, int vid)
- int create_kernvlan (char *ifname, unsigned short vid)
- int delete_kernmac (char *ifname)
- int create_kernmac (char *ifname, char *macdev, unsigned char *mac)
- int set_interface_flags (int ifindex, int set, int clear)
- int set_interface_addr (int ifindex, const unsigned char *hwaddr)
- int set_interface_name (int ifindex, const char *name)
- int interface_bind (char *iface, int protocol, int flags)
- int eui48to64 (unsigned char *mac48, unsigned char *eui64)
- int get_ip6_addrprefix (const char *iface, unsigned char *prefix)
- void randhwaddr (unsigned char *addr)
- int create_tun (const char *ifname, const unsigned char *hwaddr, int flags)
- int ifdown (const char *ifname, int flags)
- int ifup (const char *ifname, int flags)
- int ifrename (const char *oldname, const char *newname)
- int ifhwaddr (const char *ifname, unsigned char *hwaddr)
- int set_interface_ipaddr (char *ifname, char *ipaddr)

7.10.1 Detailed Description

Wrapper arround Linux libnetlink for managing network interfaces.

Definition in file interface.c.

7.11 src/iputil.c File Reference

IPv4 And IPv6 Utiliies.

```
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <linux/ip.h>
#include <linux/icmp.h>
#include <linux/tcp.h>
#include <linux/udp.h>
#include "include/dtsapp.h"
```

Data Structures

- struct pseudohdr

Enumerations

- enum ipversion { IP_PROTO_V4 = 4, IP_PROTO_V6 = 6 }

Functions

- int checkipv6mask (const char *ipaddr, const char *network, uint8_t bits)
- void ipv4tcpchecksum (uint8_t *pkt)
- void ipv4udpchecksum (uint8_t *pkt)
- void icmpchecksum (uint8_t *pkt)
- void ipv4checksum (uint8_t *pkt)
- int packetchecksumv4 (uint8_t *pkt)
- int packetchecksumv6 (uint8_t *pkt)
- int packetchecksum (uint8_t *pkt)
- const char * cidrtosn (int bitlen, const char *buf, int size)
- const char * getnetaddr (const char *ipaddr, int cidr, const char *buf, int size)
- const char * getfirstaddr (const char *ipaddr, int cidr, const char *buf, int size)
- const char * getbcaddr (const char *ipaddr, int cidr, const char *buf, int size)
- const char * getlastaddr (const char *ipaddr, int cidr, const char *buf, int size)
- uint32_t cidrcnt (int bitlen)
- int reservedip (const char *ipaddr)
- char * ipv6to4prefix (const char *ipaddr)
- int check_ipv4 (const char *ip, int cidr, const char *test)

7.11.1 Detailed Description

IPv4 And IPv6 Utiliies.

Definition in file iputil.c.

7.12 src/libnetlink/dnet_ntop.c File Reference

```
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include "utils.h"
```

Functions

- const char * dnet_ntop (int af, const void *addr, char *str, size_t len)

7.12.1 Function Documentation

7.12.1.1 const char* dnet_ntop (int *af*, const void * *addr*, char * *str*, size_t *len*)

Definition at line 97 of file dnet_ntop.c.

References AF_DECnet.

Referenced by rt_addr_n2a().

```
                                                                    {
    switch(af) {
        case AF_DECnet:
            errno = 0;
            return dnet_ntop1((struct dn_naddr *)addr, str, len);
        default
                :
            errno = EAFNOSUPPORT;
    }

    return NULL;
}
```

7.13 src/libnetlink/dnet_pton.c File Reference

```
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include "utils.h"
```

Functions

- int dnet_pton (int af, const char *src, void *addr)

7.13.1 Function Documentation

7.13.1.1 int dnet_pton (int *af*, const char * *src*, void * *addr*)

Definition at line 59 of file dnet_pton.c.

References AF_DECnet.

Referenced by get_addr_1().

```
                                                {
    int err;

    switch (af) {
        case AF_DECnet:
            errno = 0;
            err = dnet_pton1(src, (struct dn_naddr *)addr);
            break;
        default
                :
            errno = EAFNOSUPPORT;
            err = -1;
    }

    return err;
}
```

7.14 src/libnetlink/include/libnetlink.h File Reference

```
#include <string.h>
#include <asm/types.h>
#include <linux/netlink.h>
#include <linux/rtnetlink.h>
#include <linux/if_link.h>
#include <linux/if_addr.h>
#include <linux/neighbour.h>
```

Data Structures

- struct rtnl_handle
- struct rtnl_dump_filter_arg

Macros

- #define parse_rtattr_nested(tb, max, rta) (parse_rtattr((tb), (max), RTA_DATA(rta), RTA_PAYLOAD(rta)))
- #define parse_rtattr_nested_compat(tb, max, rta, data, len)
- #define NLMSG_TAIL(nmsg) ((struct rtattr *) (((char*)(nmsg)) + NLMSG_ALIGN((nmsg)->nlmsg_len)))
- #define IFA_RTA(r) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ifaddrmsg))))
- #define IFA_PAYLOAD(n) NLMSG_PAYLOAD(n,sizeof(struct ifaddrmsg))
- #define IFLA_RTA(r) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ifinfomsg))))
- #define IFLA_PAYLOAD(n) NLMSG_PAYLOAD(n,sizeof(struct ifinfomsg))
- #define NDA_RTA(r) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ndmsg))))
- #define NDA_PAYLOAD(n) NLMSG_PAYLOAD(n,sizeof(struct ndmsg))
- #define NDTA_RTA(r) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ndtmsg))))
- #define NDTA_PAYLOAD(n) NLMSG_PAYLOAD(n,sizeof(struct ndtmsg))

Typedefs

- typedef int(* rtnl_filter_t)(const struct sockaddr_nl *, struct nlmsghdr *n, void *)

Functions

- int rtnl_open (struct rtnl_handle *rth, unsigned subscriptions)
- int rtnl_open_byproto (struct rtnl_handle *rth, unsigned subscriptions, int protocol)
- void rtnl_close (struct rtnl_handle *rth)
- int rtnl_wilddump_request (struct rtnl_handle *rth, int fam, int type)

- int rtnl_dump_request (struct rtnl_handle *rth, int type, void *req, int len)
- int rtnl_dump_filter_l (struct rtnl_handle *rth, const struct rtnl_dump_filter_arg *arg)
- int rtnl_dump_filter (struct rtnl_handle *rth, rtnl_filter_t filter, void *arg)
- int rtnl_talk (struct rtnl_handle *rtnl, struct nlmsghdr *n, pid_t peer, unsigned groups, struct nlmsghdr *answer)
- int rtnl_send (struct rtnl_handle *rth, const void *buf, int)
- int rtnl_send_check (struct rtnl_handle *rth, const void *buf, int)
- int addattr (struct nlmsghdr *n, int maxlen, int type)
- int addattr8 (struct nlmsghdr *n, int maxlen, int type, __u8 data)
- int addattr16 (struct nlmsghdr *n, int maxlen, int type, __u16 data)
- int addattr32 (struct nlmsghdr *n, int maxlen, int type, __u32 data)
- int addattr64 (struct nlmsghdr *n, int maxlen, int type, __u64 data)
- int addattrstrz (struct nlmsghdr *n, int maxlen, int type, const char *data)
- int addattr_l (struct nlmsghdr *n, int maxlen, int type, const void *data, int alen)
- int addraw_l (struct nlmsghdr *n, int maxlen, const void *data, int len)
- struct rtattr * addattr_nest (struct nlmsghdr *n, int maxlen, int type)
- int addattr_nest_end (struct nlmsghdr *n, struct rtattr *nest)
- struct rtattr * addattr_nest_compat (struct nlmsghdr *n, int maxlen, int type, const void *data, int len)
- int addattr_nest_compat_end (struct nlmsghdr *n, struct rtattr *nest)
- int rta_addattr32 (struct rtattr *rta, int maxlen, int type, __u32 data)
- int rta_addattr_l (struct rtattr *rta, int maxlen, int type, const void *data, int alen)
- int parse_rtattr (struct rtattr *tb[], int max, struct rtattr *rta, int len)
- int parse_rtattr_byindex (struct rtattr *tb[], int max, struct rtattr *rta, int len)
- int __parse_rtattr_nested_compat (struct rtattr *tb[], int max, struct rtattr *rta, int len)
- int rtnl_listen (struct rtnl_handle *, rtnl_filter_t handler, void *jarg)
- int rtnl_from_file (FILE *, rtnl_filter_t handler, void *jarg)

Variables

- int rcvbuf

7.14.1 Macro Definition Documentation

7.14.1.1 #define IFA_PAYLOAD(*n*) NLMSG_PAYLOAD(n,sizeof(struct ifaddrmsg))

Definition at line 103 of file libnetlink.h.

7.14.1.2 #define IFA_RTA(*r*) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ifaddrmsg))))

Definition at line 99 of file libnetlink.h.

7.14.1.3 #define IFLA_PAYLOAD(*n*) NLMSG_PAYLOAD(n,sizeof(struct ifinfomsg))

Definition at line 111 of file libnetlink.h.

Referenced by ll_remember_index().

7.14.1.4 #define IFLA_RTA(*r*) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ifinfomsg))))

Definition at line 107 of file libnetlink.h.

Referenced by ll_remember_index().

7.14.1.5 #define NDA_PAYLOAD(*n*) NLMSG_PAYLOAD(n,sizeof(struct ndmsg))

Definition at line 119 of file libnetlink.h.

7.14.1.6 #define NDA_RTA(*r*) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ndmsg))))

Definition at line 115 of file libnetlink.h.

7.14.1.7 #define NDTA_PAYLOAD(*n*) NLMSG_PAYLOAD(n,sizeof(struct ndtmsg))

Definition at line 127 of file libnetlink.h.

7.14.1.8 #define NDTA_RTA(*r*) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ndtmsg))))

Definition at line 123 of file libnetlink.h.

7.14.1.9 #define NLMSG_TAIL(*nmsg*) ((struct rtattr *) (((char*)(nmsg)) + NLMSG_ALIGN((nmsg)->nlmsg_len)))

Definition at line 95 of file libnetlink.h.

Referenced by addattr_l(), addattr_nest(), addattr_nest_compat(), addattr_nest_compat_end(), addattr_nest_end(), addraw_l(), create_kernmac(), and create_kernvlan().

7.14.1.10 #define parse_rtattr_nested(*tb, max, rta*) (parse_rtattr((tb), (max), RTA_DATA(rta), RTA_PAYLOAD(rta)))

Definition at line 65 of file libnetlink.h.

Referenced by __parse_rtattr_nested_compat().

7.14.1.11 #define parse_rtattr_nested_compat(*tb, max, rta, data, len*)

Value:

```
({ data = RTA_PAYLOAD(rta) >= len ? RTA_DATA(rta) : NULL;   \
        __parse_rtattr_nested_compat(tb, max, rta, len); })
```

Definition at line 68 of file libnetlink.h.

7.14.2 Typedef Documentation

7.14.2.1 typedef int(* rtnl_filter_t)(const struct sockaddr_nl *, struct nlmsghdr *n, void *)

Definition at line 28 of file libnetlink.h.

7.14.3 Function Documentation

7.14.3.1 int __parse_rtattr_nested_compat (struct rtattr * *tb[],* int *max,* struct rtattr * *rta,* int *len*)

Definition at line 673 of file libnetlink.c.

```
                                                {
    if (RTA_PAYLOAD(rta) < len) {
        return -1;
    }
```

```
    if (RTA_PAYLOAD(rta) >= RTA_ALIGN(len) + sizeof(struct rtattr)) {
        rta = RTA_DATA(rta) + RTA_ALIGN(len);
        return parse_rtattr_nested(tb, max, rta);
    }
    memset(tb, 0, sizeof(struct rtattr *) * (max + 1));
    return 0;
}
```

7.14.3.2 int addattr (struct nlmsghdr * *n*, int *maxlen*, int *type*)

Definition at line 528 of file libnetlink.c.

```
                                                    {
    return addattr_l(n, maxlen, type, NULL, 0);
}
```

7.14.3.3 int addattr16 (struct nlmsghdr * *n*, int *maxlen*, int *type*, __u16 *data*)

Definition at line 536 of file libnetlink.c.

```
                                                                {
    return addattr_l(n, maxlen, type, &data, sizeof(__u16));
}
```

7.14.3.4 int addattr32 (struct nlmsghdr * *n*, int *maxlen*, int *type*, __u32 *data*)

Definition at line 540 of file libnetlink.c.

Referenced by create_kernmac(), and set_interface_ipaddr().

```
                                                                {
    return addattr_l(n, maxlen, type, &data, sizeof(__u32));
}
```

7.14.3.5 int addattr64 (struct nlmsghdr * *n*, int *maxlen*, int *type*, __u64 *data*)

Definition at line 544 of file libnetlink.c.

```
                                                                {
    return addattr_l(n, maxlen, type, &data, sizeof(__u64));
}
```

7.14.3.6 int addattr8 (struct nlmsghdr * *n*, int *maxlen*, int *type*, __u8 *data*)

Definition at line 532 of file libnetlink.c.

```
                                                               {
    return addattr_l(n, maxlen, type, &data, sizeof(__u8));
}
```

7.14.3.7 int addattr_l (struct nlmsghdr * *n*, int *maxlen*, int *type*, const void * *data*, int *alen*)

Definition at line 552 of file libnetlink.c.

Referenced by addattr(), addattr16(), addattr32(), addattr64(), addattr8(), addattr_nest(), addattr_nest_compat(), addattrstrz(), create_kernmac(), create_kernvlan(), set_interface_addr(), set_interface_ipaddr(), and set_interface_name().

```
                        {
    int len = RTA_LENGTH(alen);
    struct rtattr *rta;

    if (NLMSG_ALIGN(n->nlmsg_len) + RTA_ALIGN(len) > maxlen) {
        fprintf(stderr, "addattr_l ERROR: message exceeded bound of %d\n",
      maxlen);
        return -1;
    }
    rta = NLMSG_TAIL(n);
    rta->rta_type = type;
    rta->rta_len = len;
    memcpy(RTA_DATA(rta), data, alen);
    n->nlmsg_len = NLMSG_ALIGN(n->nlmsg_len) + RTA_ALIGN(len);
    return 0;
}
```

7.14.3.8 struct rtattr* addattr_nest (struct nlmsghdr * *n*, int *maxlen*, int *type*) [read]

Definition at line 581 of file libnetlink.c.

Referenced by addattr_nest_compat().

```
                                                                    {
    struct rtattr *nest = NLMSG_TAIL(n);

    addattr_l(n, maxlen, type, NULL, 0);
    return nest;
}
```

7.14.3.9 struct rtattr* addattr_nest_compat (struct nlmsghdr * *n*, int *maxlen*, int *type*, const void * *data*, int *len*) [read]

Definition at line 593 of file libnetlink.c.

```
                                                                        {
    struct rtattr *start = NLMSG_TAIL(n);

    addattr_l(n, maxlen, type, data, len);
    addattr_nest(n, maxlen, type);
    return start;
}
```

7.14.3.10 int addattr_nest_compat_end (struct nlmsghdr * *n*, struct rtattr * *nest*)

Definition at line 602 of file libnetlink.c.

```
                                                                    {
    struct rtattr *nest = (void *)start + NLMSG_ALIGN(start->rta_len);

    start->rta_len = (void *)NLMSG_TAIL(n) - (void *)start;
    addattr_nest_end(n, nest);
    return n->nlmsg_len;
}
```

7.14.3.11 int addattr_nest_end (struct nlmsghdr * *n*, struct rtattr * *nest*)

Definition at line 588 of file libnetlink.c.

Referenced by addattr_nest_compat_end().

```
                                                        {
    nest->rta_len = (void *)NLMSG_TAIL(n) - (void *)nest;
    return n->nlmsg_len;
}
```

7.14.3.12 int addattrstrz (struct nlmsghdr * *n*, int *maxlen*, int *type*, const char * *data*)

Definition at line 548 of file libnetlink.c.

```
                                                                    {
    return addattr_l(n, maxlen, type, str, strlen(str)+1);
}
```

7.14.3.13 int addraw_l (struct nlmsghdr * *n*, int *maxlen*, const void * *data*, int *len*)

Definition at line 569 of file libnetlink.c.

```
                                                                          {
    if (NLMSG_ALIGN(n->nlmsg_len) + NLMSG_ALIGN(len) > maxlen) {
        fprintf(stderr, "addraw_l ERROR: message exceeded bound of %d\n",maxlen
      );
        return -1;
    }

    memcpy(NLMSG_TAIL(n), data, len);
    memset((void *) NLMSG_TAIL(n) + len, 0, NLMSG_ALIGN(len) - len);
    n->nlmsg_len = NLMSG_ALIGN(n->nlmsg_len) + NLMSG_ALIGN(len);
    return 0;
}
```

7.14.3.14 int parse_rtattr (struct rtattr * *tb[]*, int *max*, struct rtattr * *rta*, int *len*)

Definition at line 643 of file libnetlink.c.

Referenced by ll_remember_index().

```
                                                                                {
    memset(tb, 0, sizeof(struct rtattr *) * (max + 1));
    while (RTA_OK(rta, len)) {
        if ((rta->rta_type <= max) && (!tb[rta->rta_type])) {
            tb[rta->rta_type] = rta;
        }
        rta = RTA_NEXT(rta,len);
    }
    if (len) {
        fprintf(stderr, "!!!Deficit %d, rta_len=%d\n", len, rta->rta_len);
    }
    return 0;
}
```

7.14.3.15 int parse_rtattr_byindex (struct rtattr * *tb[]*, int *max*, struct rtattr * *rta*, int *len*)

Definition at line 657 of file libnetlink.c.

```
          {
    int i = 0;
```

```
    memset(tb, 0, sizeof(struct rtattr *) * max);
    while (RTA_OK(rta, len)) {
        if (rta->rta_type <= max && i < max) {
            tb[i++] = rta;
        }
        rta = RTA_NEXT(rta,len);
    }
    if (len) {
        fprintf(stderr, "!!!Deficit %d, rta_len=%d\n", len, rta->rta_len);
    }
    return i;
}
```

7.14.3.16 int rta_addattr32 (struct rtattr * *rta*, int *maxlen*, int *type*, __u32 *data*)

Definition at line 610 of file libnetlink.c.

```
{
    int len = RTA_LENGTH(4);
    struct rtattr *subrta;

    if (RTA_ALIGN(rta->rta_len) + len > maxlen) {
        fprintf(stderr,"rta_addattr32: Error! max allowed bound %d exceeded\n",
      maxlen);
        return -1;
    }
    subrta = (struct rtattr *)(((char *)rta) + RTA_ALIGN(rta->rta_len));
    subrta->rta_type = type;
    subrta->rta_len = len;
    memcpy(RTA_DATA(subrta), &data, 4);
    rta->rta_len = NLMSG_ALIGN(rta->rta_len) + len;
    return 0;
}
```

7.14.3.17 int rta_addattr_l (struct rtattr * *rta*, int *maxlen*, int *type*, const void * *data*, int *alen*)

Definition at line 626 of file libnetlink.c.

```
{
    struct rtattr *subrta;
    int len = RTA_LENGTH(alen);

    if (RTA_ALIGN(rta->rta_len) + RTA_ALIGN(len) > maxlen) {
        fprintf(stderr,"rta_addattr_l: Error! max allowed bound %d exceeded\n",
      maxlen);
        return -1;
    }
    subrta = (struct rtattr *)(((char *)rta) + RTA_ALIGN(rta->rta_len));
    subrta->rta_type = type;
    subrta->rta_len = len;
    memcpy(RTA_DATA(subrta), data, alen);
    rta->rta_len = NLMSG_ALIGN(rta->rta_len) + RTA_ALIGN(len);
    return 0;
}
```

7.14.3.18 void rtnl_close (struct rtnl_handle * *rth*)

Definition at line 30 of file libnetlink.c.

```
{
    if (rth->fd >= 0) {
        close(rth->fd);
        rth->fd = -1;
    }
}
```

7.14.3.19 int rtnl_dump_filter (struct rtnl_handle * *rth*, rtnl_filter_t *filter*, void * *arg*)

Definition at line 264 of file libnetlink.c.

Referenced by ll_init_map().

```
                                {
    const struct rtnl_dump_filter_arg a[2] = {
        { .filter = filter, .arg1 = arg1, },
        { .filter = NULL,   .arg1 = NULL, },
    };

    return rtnl_dump_filter_l(rth, a);
}
```

7.14.3.20 int rtnl_dump_filter_l (struct rtnl_handle * *rth*, const struct rtnl_dump_filter_arg * *arg*)

Definition at line 175 of file libnetlink.c.

Referenced by rtnl_dump_filter().

```
                                                                {
    struct sockaddr_nl nladdr;
    struct iovec iov;
    struct msghdr msg = {
        .msg_name = &nladdr,
        .msg_namelen = sizeof(nladdr),
        .msg_iov = &iov,
        .msg_iovlen = 1,
    };
    char buf[16384];

    iov.iov_base = buf;
    while (1) {
        int status;
        const struct rtnl_dump_filter_arg *a;
        int found_done = 0;
        int msglen = 0;

        iov.iov_len = sizeof(buf);
        status = recvmsg(rth->fd, &msg, 0);

        if (status < 0) {
            if (errno == EINTR || errno == EAGAIN) {
                continue;
            }
            fprintf(stderr, "netlink receive error %s (%d)\n",
                    strerror(errno), errno);
            return -1;
        }

        if (status == 0) {
            fprintf(stderr, "EOF on netlink\n");
            return -1;
        }

        for (a = arg; a->filter; a++) {
            struct nlmsghdr *h = (struct nlmsghdr *)buf;
            msglen = status;

            while (NLMSG_OK(h, msglen)) {
                int err;

                if (nladdr.nl_pid != 0 ||
                        h->nlmsg_pid != rth->local.nl_pid ||
                        h->nlmsg_seq != rth->dump) {
                    goto skip_it;
                }

                if (h->nlmsg_type == NLMSG_DONE) {
                    found_done = 1;
                    break; /* process next filter */
                }
                if (h->nlmsg_type == NLMSG_ERROR) {
                    struct nlmsgerr *err = (struct nlmsgerr *)NLMSG_DATA(h);
                    if (h->nlmsg_len < NLMSG_LENGTH(sizeof(struct nlmsgerr))) {
                        fprintf(stderr,
                                "ERROR truncated\n");
                    } else {
                        errno = -err->error;
```

```
                        perror("RTNETLINK answers");
                    }
                    return -1;
                }
                err = a->filter(&nladdr, h, a->arg1);
                if (err < 0) {
                    return err;
                }

skip_it:
                h = NLMSG_NEXT(h, msglen);
            }
        }

        if (found_done) {
            return 0;
        }

        if (msg.msg_flags & MSG_TRUNC) {
            fprintf(stderr, "Message truncated\n");
            continue;
        }
        if (msglen) {
            fprintf(stderr, "!!!Remnant of size %d\n", msglen);
            exit(1);
        }
    }
}
```

7.14.3.21 int rtnl_dump_request (struct rtnl_handle * *rth*, int *type*, void * *req*, int *len*)

Definition at line 152 of file libnetlink.c.

```
                                                                                {
    struct nlmsghdr nlh;
    struct sockaddr_nl nladdr = { .nl_family = AF_NETLINK };
    struct iovec iov[2] = {
        { .iov_base = &nlh, .iov_len = sizeof(nlh) },
        { .iov_base = req, .iov_len = len }
    };
    struct msghdr msg = {
        .msg_name = &nladdr,
        .msg_namelen =  sizeof(nladdr),
        .msg_iov = iov,
        .msg_iovlen = 2,
    };

    nlh.nlmsg_len = NLMSG_LENGTH(len);
    nlh.nlmsg_type = type;
    nlh.nlmsg_flags = NLM_F_DUMP|NLM_F_REQUEST;
    nlh.nlmsg_pid = 0;
    nlh.nlmsg_seq = rth->dump = ++rth->seq;

    return sendmsg(rth->fd, &msg, 0);
}
```

7.14.3.22 int rtnl_from_file (FILE * , rtnl_filter_t *handler*, void * *jarg*)

Definition at line 472 of file libnetlink.c.

```
                                {
    int status;
    struct sockaddr_nl nladdr;
    char   buf[8192];
    struct nlmsghdr *h = (void *)buf;

    memset(&nladdr, 0, sizeof(nladdr));
    nladdr.nl_family = AF_NETLINK;
    nladdr.nl_pid = 0;
    nladdr.nl_groups = 0;

    while (1) {
        int err, len;
        int l;

        status = fread(&buf, 1, sizeof(*h), rtnl);
```

```
        if (status < 0) {
            if (errno == EINTR) {
                continue;
            }
            perror("rtnl_from_file: fread");
            return -1;
        }
        if (status == 0) {
            return 0;
        }

        len = h->nlmsg_len;
        l = len - sizeof(*h);

        if (l<0 || len>sizeof(buf)) {
            fprintf(stderr, "!!!malformed message: len=%d @%lu\n",
                    len, ftell(rtnl));
            return -1;
        }

        status = fread(NLMSG_DATA(h), 1, NLMSG_ALIGN(l), rtnl);

        if (status < 0) {
            perror("rtnl_from_file: fread");
            return -1;
        }
        if (status < l) {
            fprintf(stderr, "rtnl-from_file: truncated message\n");
            return -1;
        }

        err = handler(&nladdr, h, jarg);
        if (err < 0) {
            return err;
        }
    }
}
```

7.14.3.23 int rtnl_listen (struct rtnl_handle * , rtnl_filter_t *handler,* void * *jarg*)

Definition at line 395 of file libnetlink.c.

```
                            {
    int status;
    struct nlmsghdr *h;
    struct sockaddr_nl nladdr;
    struct iovec iov;
    struct msghdr msg = {
        .msg_name = &nladdr,
        .msg_namelen = sizeof(nladdr),
        .msg_iov = &iov,
        .msg_iovlen = 1,
    };
    char   buf[8192];

    memset(&nladdr, 0, sizeof(nladdr));
    nladdr.nl_family = AF_NETLINK;
    nladdr.nl_pid = 0;
    nladdr.nl_groups = 0;

    iov.iov_base = buf;
    while (1) {
        iov.iov_len = sizeof(buf);
        status = recvmsg(rtnl->fd, &msg, 0);

        if (status < 0) {
            if (errno == EINTR || errno == EAGAIN) {
                continue;
            }
            fprintf(stderr, "netlink receive error %s (%d)\n",
                    strerror(errno), errno);
            if (errno == ENOBUFS) {
                continue;
            }
            return -1;
        }
        if (status == 0) {
            fprintf(stderr, "EOF on netlink\n");
            return -1;
        }
        if (msg.msg_namelen != sizeof(nladdr)) {
            fprintf(stderr, "Sender address length == %d\n", msg.msg_namelen);
```

```
            exit(1);
        }
        for (h = (struct nlmsghdr *)buf; status >= sizeof(*h); ) {
            int err;
            int len = h->nlmsg_len;
            int l = len - sizeof(*h);

            if (l<0 || len>status) {
                if (msg.msg_flags & MSG_TRUNC) {
                    fprintf(stderr, "Truncated message\n");
                    return -1;
                }
                fprintf(stderr, "!!!malformed message: len=%d\n", len);
                exit(1);
            }

            err = handler(&nladdr, h, jarg);
            if (err < 0) {
                return err;
            }

            status -= NLMSG_ALIGN(len);
            h = (struct nlmsghdr *)((char *)h + NLMSG_ALIGN(len));
        }
        if (msg.msg_flags & MSG_TRUNC) {
            fprintf(stderr, "Message truncated\n");
            continue;
        }
        if (status) {
            fprintf(stderr, "!!!Remnant of size %d\n", status);
            exit(1);
        }
    }
}
```

7.14.3.24 int rtnl_open (struct rtnl_handle * *rth*, unsigned *subscriptions*)

Definition at line 85 of file libnetlink.c.

```
                                                                    {
    return rtnl_open_byproto(rth, subscriptions, NETLINK_ROUTE
      );
}
```

7.14.3.25 int rtnl_open_byproto (struct rtnl_handle * *rth*, unsigned *subscriptions*, int *protocol*)

Definition at line 37 of file libnetlink.c.

Referenced by rtnl_open().

```
                                    {
    socklen_t addr_len;
    int sndbuf = 32768;

    memset(rth, 0, sizeof(*rth));

    rth->fd = socket(AF_NETLINK, SOCK_RAW, protocol);
    if (rth->fd < 0) {
        perror("Cannot open netlink socket");
        return -1;
    }

    if (setsockopt(rth->fd,SOL_SOCKET,SO_SNDBUF,&sndbuf,sizeof(sndbuf)) < 0)
      {
        perror("SO_SNDBUF");
        return -1;
    }

    if (setsockopt(rth->fd,SOL_SOCKET,SO_RCVBUF,&rcvbuf,sizeof(rcvbuf
      )) < 0) {
        perror("SO_RCVBUF");
        return -1;
    }

    memset(&rth->local, 0, sizeof(rth->local));
    rth->local.nl_family = AF_NETLINK;
    rth->local.nl_groups = subscriptions;
```

```
    if (bind(rth->fd, (struct sockaddr *)&rth->local, sizeof(rth->local
      )) < 0) {
        perror("Cannot bind netlink socket");
        return -1;
    }
    addr_len = sizeof(rth->local);
    if (getsockname(rth->fd, (struct sockaddr *)&rth->local, &addr_len)
      < 0) {
        perror("Cannot getsockname");
        return -1;
    }
    if (addr_len != sizeof(rth->local)) {
        fprintf(stderr, "Wrong address length %d\n", addr_len);
        return -1;
    }
    if (rth->local.nl_family != AF_NETLINK) {
        fprintf(stderr, "Wrong address family %d\n", rth->local.nl_family)
      ;
        return -1;
    }
    rth->seq = time(NULL);
    return 0;
}
```

7.14.3.26 int rtnl_send (struct rtnl_handle * *rth*, const void * *buf*, int)

Definition at line 113 of file libnetlink.c.

```
{
    return send(rth->fd, buf, len, 0);
}
```

7.14.3.27 int rtnl_send_check (struct rtnl_handle * *rth*, const void * *buf*, int)

Definition at line 117 of file libnetlink.c.

```
{
    struct nlmsghdr *h;
    int status;
    char resp[1024];

    status = send(rth->fd, buf, len, 0);
    if (status < 0) {
        return status;
    }

    /* Check for immediate errors */
    status = recv(rth->fd, resp, sizeof(resp), MSG_DONTWAIT|MSG_PEEK);
    if (status < 0) {
        if (errno == EAGAIN) {
            return 0;
        }
        return -1;
    }

    for (h = (struct nlmsghdr *)resp; NLMSG_OK(h, status);
            h = NLMSG_NEXT(h, status)) {
        if (h->nlmsg_type == NLMSG_ERROR) {
            struct nlmsgerr *err = (struct nlmsgerr *)NLMSG_DATA(h);
            if (h->nlmsg_len < NLMSG_LENGTH(sizeof(struct nlmsgerr))) {
                fprintf(stderr, "ERROR truncated\n");
            } else {
                errno = -err->error;
            }
            return -1;
        }
    }

    return 0;
}
```

7.14.3.28 int rtnl_talk (struct rtnl_handle * *rtnl*, struct nlmsghdr * *n*, pid_t *peer*, unsigned *groups*, struct nlmsghdr * *answer*)

Definition at line 275 of file libnetlink.c.

Referenced by create_kernmac(), create_kernvlan(), set_interface_addr(), set_interface_flags(), set_interface_ipaddr(), and set_interface_name().

```
                                                {
    int status;
    unsigned seq;
    struct nlmsghdr *h;
    struct sockaddr_nl nladdr;
    struct iovec iov = {
        .iov_base = (void *) n,
        .iov_len = n->nlmsg_len
    };
    struct msghdr msg = {
        .msg_name = &nladdr,
        .msg_namelen = sizeof(nladdr),
        .msg_iov = &iov,
        .msg_iovlen = 1,
    };
    char   buf[16384];

    memset(&nladdr, 0, sizeof(nladdr));
    nladdr.nl_family = AF_NETLINK;
    nladdr.nl_pid = peer;
    nladdr.nl_groups = groups;

    n->nlmsg_seq = seq = ++rtnl->seq;

    if (answer == NULL) {
        n->nlmsg_flags |= NLM_F_ACK;
    }

    status = sendmsg(rtnl->fd, &msg, 0);

    if (status < 0) {
        perror("Cannot talk to rtnetlink");
        return -1;
    }

    memset(buf,0,sizeof(buf));

    iov.iov_base = buf;

    while (1) {
        iov.iov_len = sizeof(buf);
        status = recvmsg(rtnl->fd, &msg, 0);

        if (status < 0) {
            if (errno == EINTR || errno == EAGAIN) {
                continue;
            }
            fprintf(stderr, "netlink receive error %s (%d)\n",
                    strerror(errno), errno);
            return -1;
        }
        if (status == 0) {
            fprintf(stderr, "EOF on netlink\n");
            return -1;
        }
        if (msg.msg_namelen != sizeof(nladdr)) {
            fprintf(stderr, "sender address length == %d\n", msg.msg_namelen);
            exit(1);
        }
        for (h = (struct nlmsghdr *)buf; status >= sizeof(*h); ) {
            int len = h->nlmsg_len;
            int l = len - sizeof(*h);

            if (l < 0 || len>status) {
                if (msg.msg_flags & MSG_TRUNC) {
                    fprintf(stderr, "Truncated message\n");
                    return -1;
                }
                fprintf(stderr, "!!!malformed message: len=%d\n", len);
                exit(1);
            }

            if (nladdr.nl_pid != peer ||
                    h->nlmsg_pid != rtnl->local.nl_pid ||
                    h->nlmsg_seq != seq) {
                /* Don't forget to skip that message. */
```

```
                status -= NLMSG_ALIGN(len);
                h = (struct nlmsghdr *)((char *)h + NLMSG_ALIGN(len));
                continue;
            }

            if (h->nlmsg_type == NLMSG_ERROR) {
                struct nlmsgerr *err = (struct nlmsgerr *)NLMSG_DATA(h);
                if (l < sizeof(struct nlmsgerr)) {
                    fprintf(stderr, "ERROR truncated\n");
                } else {
                    if (!err->error) {
                        if (answer) {
                            memcpy(answer, h, h->nlmsg_len);
                        }
                        return 0;
                    }

                    fprintf(stderr, "RTNETLINK answers: %s\n", strerror(-err->
      error));
                    errno = -err->error;
                }
                return -1;
            }
            if (answer) {
                memcpy(answer, h, h->nlmsg_len);
                return 0;
            }

            fprintf(stderr, "Unexpected reply!!!\n");

            status -= NLMSG_ALIGN(len);
            h = (struct nlmsghdr *)((char *)h + NLMSG_ALIGN(len));
        }
        if (msg.msg_flags & MSG_TRUNC) {
            fprintf(stderr, "Message truncated\n");
            continue;
        }
        if (status) {
            fprintf(stderr, "!!!Remnant of size %d\n", status);
            exit(1);
        }
    }
}
```

7.14.3.29 int rtnl_wilddump_request (struct rtnl_handle * *rth*, int *fam*, int *type*)

Definition at line 89 of file libnetlink.c.

Referenced by ll_init_map().

```
                                                                        {
    struct {
        struct nlmsghdr nlh;
        struct rtgenmsg g;
        __u16 align_rta;    /* attribute has to be 32bit aligned */
        struct rtattr ext_req;
        __u32 ext_filter_mask;
    } req;

    memset(&req, 0, sizeof(req));
    req.nlh.nlmsg_len = sizeof(req);
    req.nlh.nlmsg_type = type;
    req.nlh.nlmsg_flags = NLM_F_DUMP|NLM_F_REQUEST;
    req.nlh.nlmsg_pid = 0;
    req.nlh.nlmsg_seq = rth->dump = ++rth->seq;
    req.g.rtgen_family = family;

    req.ext_req.rta_type = IFLA_EXT_MASK;
    req.ext_req.rta_len = RTA_LENGTH(sizeof(__u32));
    req.ext_filter_mask = RTEXT_FILTER_VF;

    return send(rth->fd, (void *)&req, sizeof(req), 0);
}
```

7.14.4 Variable Documentation

7.14.4.1 int rcvbuf

Definition at line 28 of file libnetlink.c.

Referenced by rtnl_open_byproto().

7.15 src/libnetlink/libnetlink.h File Reference

```
#include <string.h>
#include <asm/types.h>
#include <linux/netlink.h>
#include <linux/rtnetlink.h>
#include <linux/if_link.h>
#include <linux/if_addr.h>
#include <linux/neighbour.h>
```

Data Structures

- struct rtnl_handle
- struct rtnl_dump_filter_arg

Macros

- #define parse_rtattr_nested(tb, max, rta) (parse_rtattr((tb), (max), RTA_DATA(rta), RTA_PAYLOAD(rta)))
- #define parse_rtattr_nested_compat(tb, max, rta, data, len)
- #define NLMSG_TAIL(nmsg) ((struct rtattr *) (((void *) (nmsg)) + NLMSG_ALIGN((nmsg)->nlmsg_len)))
- #define IFA_RTA(r) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ifaddrmsg))))
- #define IFA_PAYLOAD(n) NLMSG_PAYLOAD(n,sizeof(struct ifaddrmsg))
- #define IFLA_RTA(r) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ifinfomsg))))
- #define IFLA_PAYLOAD(n) NLMSG_PAYLOAD(n,sizeof(struct ifinfomsg))
- #define NDA_RTA(r) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ndmsg))))
- #define NDA_PAYLOAD(n) NLMSG_PAYLOAD(n,sizeof(struct ndmsg))
- #define NDTA_RTA(r) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ndtmsg))))
- #define NDTA_PAYLOAD(n) NLMSG_PAYLOAD(n,sizeof(struct ndtmsg))

Typedefs

- typedef int(* rtnl_filter_t)(const struct sockaddr_nl *, struct nlmsghdr *n, void *)

Functions

- int rtnl_open (struct rtnl_handle *rth, unsigned subscriptions)
- int rtnl_open_byproto (struct rtnl_handle *rth, unsigned subscriptions, int protocol)
- void rtnl_close (struct rtnl_handle *rth)
- int rtnl_wilddump_request (struct rtnl_handle *rth, int fam, int type)
- int rtnl_dump_request (struct rtnl_handle *rth, int type, void *req, int len)
- int rtnl_dump_filter_l (struct rtnl_handle *rth, const struct rtnl_dump_filter_arg *arg)
- int rtnl_dump_filter (struct rtnl_handle *rth, rtnl_filter_t filter, void *arg)
- int rtnl_talk (struct rtnl_handle *rtnl, struct nlmsghdr *n, pid_t peer, unsigned groups, struct nlmsghdr *answer)
- int rtnl_send (struct rtnl_handle *rth, const void *buf, int)

- int rtnl_send_check (struct rtnl_handle *rth, const void *buf, int)
- int addattr (struct nlmsghdr *n, int maxlen, int type)
- int addattr8 (struct nlmsghdr *n, int maxlen, int type, __u8 data)
- int addattr16 (struct nlmsghdr *n, int maxlen, int type, __u16 data)
- int addattr32 (struct nlmsghdr *n, int maxlen, int type, __u32 data)
- int addattr64 (struct nlmsghdr *n, int maxlen, int type, __u64 data)
- int addattrstrz (struct nlmsghdr *n, int maxlen, int type, const char *data)
- int addattr_l (struct nlmsghdr *n, int maxlen, int type, const void *data, int alen)
- int addraw_l (struct nlmsghdr *n, int maxlen, const void *data, int len)
- struct rtattr * addattr_nest (struct nlmsghdr *n, int maxlen, int type)
- int addattr_nest_end (struct nlmsghdr *n, struct rtattr *nest)
- struct rtattr * addattr_nest_compat (struct nlmsghdr *n, int maxlen, int type, const void *data, int len)
- int addattr_nest_compat_end (struct nlmsghdr *n, struct rtattr *nest)
- int rta_addattr32 (struct rtattr *rta, int maxlen, int type, __u32 data)
- int rta_addattr_l (struct rtattr *rta, int maxlen, int type, const void *data, int alen)
- int parse_rtattr (struct rtattr *tb[], int max, struct rtattr *rta, int len)
- int parse_rtattr_byindex (struct rtattr *tb[], int max, struct rtattr *rta, int len)
- int __parse_rtattr_nested_compat (struct rtattr *tb[], int max, struct rtattr *rta, int len)
- int rtnl_listen (struct rtnl_handle *, rtnl_filter_t handler, void *jarg)
- int rtnl_from_file (FILE *, rtnl_filter_t handler, void *jarg)

Variables

- int rcvbuf

7.15.1 Macro Definition Documentation

7.15.1.1 #define IFA_PAYLOAD(*n*) NLMSG_PAYLOAD(n,sizeof(struct ifaddrmsg))

Definition at line 103 of file libnetlink.h.

7.15.1.2 #define IFA_RTA(*r*) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ifaddrmsg))))

Definition at line 99 of file libnetlink.h.

7.15.1.3 #define IFLA_PAYLOAD(*n*) NLMSG_PAYLOAD(n,sizeof(struct ifinfomsg))

Definition at line 111 of file libnetlink.h.

7.15.1.4 #define IFLA_RTA(*r*) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ifinfomsg))))

Definition at line 107 of file libnetlink.h.

7.15.1.5 #define NDA_PAYLOAD(*n*) NLMSG_PAYLOAD(n,sizeof(struct ndmsg))

Definition at line 119 of file libnetlink.h.

7.15.1.6 #define NDA_RTA(*r*) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ndmsg))))

Definition at line 115 of file libnetlink.h.

7.15.1.7 #define NDTA_PAYLOAD(*n*) NLMSG_PAYLOAD(n,sizeof(struct ndtmsg))

Definition at line 127 of file libnetlink.h.

7.15.1.8 #define NDTA_RTA(*r*) ((struct rtattr*)(((char*)(r)) + NLMSG_ALIGN(sizeof(struct ndtmsg))))

Definition at line 123 of file libnetlink.h.

7.15.1.9 #define NLMSG_TAIL(*nmsg*) ((struct rtattr *) (((void *) (nmsg)) + NLMSG_ALIGN((nmsg)->nlmsg_len)))

Definition at line 95 of file libnetlink.h.

7.15.1.10 #define parse_rtattr_nested(*tb, max, rta*) (parse_rtattr((tb), (max), RTA_DATA(rta), RTA_PAYLOAD(rta)))

Definition at line 65 of file libnetlink.h.

7.15.1.11 #define parse_rtattr_nested_compat(*tb, max, rta, data, len*)

Value:

```
({ data = RTA_PAYLOAD(rta) >= len ? RTA_DATA(rta) : NULL;   \
        __parse_rtattr_nested_compat(tb, max, rta, len); })
```

Definition at line 68 of file libnetlink.h.

7.15.2 Typedef Documentation

7.15.2.1 typedef int(* rtnl_filter_t)(const struct sockaddr_nl *, struct nlmsghdr *n, void *)

Definition at line 28 of file libnetlink.h.

7.15.3 Function Documentation

7.15.3.1 int __parse_rtattr_nested_compat (struct rtattr * *tb[]*, int *max*, struct rtattr * *rta*, int *len*)

Definition at line 673 of file libnetlink.c.

References parse_rtattr_nested.

```
                                        {
    if (RTA_PAYLOAD(rta) < len) {
        return -1;
    }
    if (RTA_PAYLOAD(rta) >= RTA_ALIGN(len) + sizeof(struct rtattr)) {
        rta = RTA_DATA(rta) + RTA_ALIGN(len);
        return parse_rtattr_nested(tb, max, rta);
    }
    memset(tb, 0, sizeof(struct rtattr *) * (max + 1));
    return 0;
}
```

7.15.3.2 int addattr (struct nlmsghdr * *n*, int *maxlen*, int *type*)

Definition at line 528 of file libnetlink.c.

References addattr_l().

```
                                                {
    return addattr_l(n, maxlen, type, NULL, 0);
}
```

7.15.3.3 int addattr16 (struct nlmsghdr * *n*, int *maxlen*, int *type*, __u16 *data*)

Definition at line 536 of file libnetlink.c.

References addattr_l().

```
                                                                {
    return addattr_l(n, maxlen, type, &data, sizeof(__u16));
}
```

7.15.3.4 int addattr32 (struct nlmsghdr * *n*, int *maxlen*, int *type*, __u32 *data*)

Definition at line 540 of file libnetlink.c.

References addattr_l().

```
                                                                {
    return addattr_l(n, maxlen, type, &data, sizeof(__u32));
}
```

7.15.3.5 int addattr64 (struct nlmsghdr * *n*, int *maxlen*, int *type*, __u64 *data*)

Definition at line 544 of file libnetlink.c.

References addattr_l().

```
                                                                {
    return addattr_l(n, maxlen, type, &data, sizeof(__u64));
}
```

7.15.3.6 int addattr8 (struct nlmsghdr * *n*, int *maxlen*, int *type*, __u8 *data*)

Definition at line 532 of file libnetlink.c.

References addattr_l().

```
                                                               {
    return addattr_l(n, maxlen, type, &data, sizeof(__u8));
}
```

7.15.3.7 int addattr_l (struct nlmsghdr * *n*, int *maxlen*, int *type*, const void * *data*, int *alen*)

Definition at line 552 of file libnetlink.c.

References NLMSG_TAIL.

```
                                {
    int len = RTA_LENGTH(alen);
    struct rtattr *rta;

    if (NLMSG_ALIGN(n->nlmsg_len) + RTA_ALIGN(len) > maxlen) {
        fprintf(stderr, "addattr_l ERROR: message exceeded bound of %d\n",
      maxlen);
        return -1;
    }
    rta = NLMSG_TAIL(n);
```

```
    rta->rta_type = type;
    rta->rta_len = len;
    memcpy(RTA_DATA(rta), data, alen);
    n->nlmsg_len = NLMSG_ALIGN(n->nlmsg_len) + RTA_ALIGN(len);
    return 0;
}
```

7.15.3.8 struct rtattr* addattr_nest (struct nlmsghdr * *n*, int *maxlen*, int *type*) [read]

Definition at line 581 of file libnetlink.c.

References addattr_l(), and NLMSG_TAIL.

```
{
    struct rtattr *nest = NLMSG_TAIL(n);

    addattr_l(n, maxlen, type, NULL, 0);
    return nest;
}
```

7.15.3.9 struct rtattr* addattr_nest_compat (struct nlmsghdr * *n*, int *maxlen*, int *type*, const void * *data*, int *len*) [read]

Definition at line 593 of file libnetlink.c.

References addattr_l(), addattr_nest(), and NLMSG_TAIL.

```
{
    struct rtattr *start = NLMSG_TAIL(n);

    addattr_l(n, maxlen, type, data, len);
    addattr_nest(n, maxlen, type);
    return start;
}
```

7.15.3.10 int addattr_nest_compat_end (struct nlmsghdr * *n*, struct rtattr * *nest*)

Definition at line 602 of file libnetlink.c.

References addattr_nest_end(), and NLMSG_TAIL.

```
{
    struct rtattr *nest = (void *)start + NLMSG_ALIGN(start->rta_len);

    start->rta_len = (void *)NLMSG_TAIL(n) - (void *)start;
    addattr_nest_end(n, nest);
    return n->nlmsg_len;
}
```

7.15.3.11 int addattr_nest_end (struct nlmsghdr * *n*, struct rtattr * *nest*)

Definition at line 588 of file libnetlink.c.

References NLMSG_TAIL.

```
{
    nest->rta_len = (void *)NLMSG_TAIL(n) - (void *)nest;
    return n->nlmsg_len;
}
```

7.15.3.12 int addattrstrz (struct nlmsghdr * *n*, int *maxlen*, int *type*, const char * *data*)

Definition at line 548 of file libnetlink.c.

References addattr_l().

```
{
    return addattr_l(n, maxlen, type, str, strlen(str)+1);
}
```

7.15.3.13 int addraw_l (struct nlmsghdr * *n*, int *maxlen*, const void * *data*, int *len*)

Definition at line 569 of file libnetlink.c.

References NLMSG_TAIL.

```
{
    if (NLMSG_ALIGN(n->nlmsg_len) + NLMSG_ALIGN(len) > maxlen) {
        fprintf(stderr, "addraw_l ERROR: message exceeded bound of %d\n",maxlen
      );
        return -1;
    }

    memcpy(NLMSG_TAIL(n), data, len);
    memset((void *) NLMSG_TAIL(n) + len, 0, NLMSG_ALIGN(len) - len);
    n->nlmsg_len = NLMSG_ALIGN(n->nlmsg_len) + NLMSG_ALIGN(len);
    return 0;
}
```

7.15.3.14 int parse_rtattr (struct rtattr * *tb[]*, int *max*, struct rtattr * *rta*, int *len*)

Definition at line 643 of file libnetlink.c.

```
{
    memset(tb, 0, sizeof(struct rtattr *) * (max + 1));
    while (RTA_OK(rta, len)) {
        if ((rta->rta_type <= max) && (!tb[rta->rta_type])) {
            tb[rta->rta_type] = rta;
        }
        rta = RTA_NEXT(rta,len);
    }
    if (len) {
        fprintf(stderr, "!!!Deficit %d, rta_len=%d\n", len, rta->rta_len);
    }
    return 0;
}
```

7.15.3.15 int parse_rtattr_byindex (struct rtattr * *tb[]*, int *max*, struct rtattr * *rta*, int *len*)

Definition at line 657 of file libnetlink.c.

```
{
    int i = 0;

    memset(tb, 0, sizeof(struct rtattr *) * max);
    while (RTA_OK(rta, len)) {
        if (rta->rta_type <= max && i < max) {
            tb[i++] = rta;
        }
        rta = RTA_NEXT(rta,len);
    }
    if (len) {
        fprintf(stderr, "!!!Deficit %d, rta_len=%d\n", len, rta->rta_len);
    }
    return i;
}
```

7.15.3.16 int rta_addattr32 (struct rtattr * *rta*, int *maxlen*, int *type*, __u32 *data*)

Definition at line 610 of file libnetlink.c.

```
                                                                        {
    int len = RTA_LENGTH(4);
    struct rtattr *subrta;

    if (RTA_ALIGN(rta->rta_len) + len > maxlen) {
        fprintf(stderr,"rta_addattr32: Error! max allowed bound %d exceeded\n",
      maxlen);
        return -1;
    }
    subrta = (struct rtattr *)(((char *)rta) + RTA_ALIGN(rta->rta_len));
    subrta->rta_type = type;
    subrta->rta_len = len;
    memcpy(RTA_DATA(subrta), &data, 4);
    rta->rta_len = NLMSG_ALIGN(rta->rta_len) + len;
    return 0;
}
```

7.15.3.17 int rta_addattr_l (struct rtattr * *rta*, int *maxlen*, int *type*, const void * *data*, int *alen*)

Definition at line 626 of file libnetlink.c.

```
                                                {
    struct rtattr *subrta;
    int len = RTA_LENGTH(alen);

    if (RTA_ALIGN(rta->rta_len) + RTA_ALIGN(len) > maxlen) {
        fprintf(stderr,"rta_addattr_l: Error! max allowed bound %d exceeded\n",
      maxlen);
        return -1;
    }
    subrta = (struct rtattr *)(((char *)rta) + RTA_ALIGN(rta->rta_len));
    subrta->rta_type = type;
    subrta->rta_len = len;
    memcpy(RTA_DATA(subrta), data, alen);
    rta->rta_len = NLMSG_ALIGN(rta->rta_len) + RTA_ALIGN(len);
    return 0;
}
```

7.15.3.18 void rtnl_close (struct rtnl_handle * *rth*)

Definition at line 30 of file libnetlink.c.

References rtnl_handle::fd.

```
                                              {
    if (rth->fd >= 0) {
        close(rth->fd);
        rth->fd = -1;
    }
}
```

7.15.3.19 int rtnl_dump_filter (struct rtnl_handle * *rth*, rtnl_filter_t *filter*, void * *arg*)

Definition at line 264 of file libnetlink.c.

References rtnl_dump_filter_arg::arg1, rtnl_dump_filter_arg::filter, and rtnl_dump_filter_l().

```
                                    {
    const struct rtnl_dump_filter_arg a[2] = {
        { .filter = filter, .arg1 = arg1, },
        { .filter = NULL,   .arg1 = NULL, },
    };

    return rtnl_dump_filter_l(rth, a);
}
```

7.15.3.20 int rtnl_dump_filter_l (struct rtnl_handle * *rth*, const struct rtnl_dump_filter_arg * *arg*)

Definition at line 175 of file libnetlink.c.

References rtnl_dump_filter_arg::arg1, rtnl_handle::dump, rtnl_handle::fd, rtnl_dump_filter_arg::filter, and rtnl_handle::local.

```
                                                                {
    struct sockaddr_nl nladdr;
    struct iovec iov;
    struct msghdr msg = {
        .msg_name = &nladdr,
        .msg_namelen = sizeof(nladdr),
        .msg_iov = &iov,
        .msg_iovlen = 1,
    };
    char buf[16384];

    iov.iov_base = buf;
    while (1) {
        int status;
        const struct rtnl_dump_filter_arg *a;
        int found_done = 0;
        int msglen = 0;

        iov.iov_len = sizeof(buf);
        status = recvmsg(rth->fd, &msg, 0);

        if (status < 0) {
            if (errno == EINTR || errno == EAGAIN) {
                continue;
            }
            fprintf(stderr, "netlink receive error %s (%d)\n",
                    strerror(errno), errno);
            return -1;
        }

        if (status == 0) {
            fprintf(stderr, "EOF on netlink\n");
            return -1;
        }

        for (a = arg; a->filter; a++) {
            struct nlmsghdr *h = (struct nlmsghdr *)buf;
            msglen = status;

            while (NLMSG_OK(h, msglen)) {
                int err;

                if (nladdr.nl_pid != 0 ||
                        h->nlmsg_pid != rth->local.nl_pid ||
                        h->nlmsg_seq != rth->dump) {
                    goto skip_it;
                }

                if (h->nlmsg_type == NLMSG_DONE) {
                    found_done = 1;
                    break; /* process next filter */
                }
                if (h->nlmsg_type == NLMSG_ERROR) {
                    struct nlmsgerr *err = (struct nlmsgerr *)NLMSG_DATA(h);
                    if (h->nlmsg_len < NLMSG_LENGTH(sizeof(struct nlmsgerr))) {
                        fprintf(stderr,
                                "ERROR truncated\n");
                    } else {
                        errno = -err->error;
                        perror("RTNETLINK answers");
                    }
                    return -1;
                }
                err = a->filter(&nladdr, h, a->arg1);
                if (err < 0) {
                    return err;
                }

skip_it:
                h = NLMSG_NEXT(h, msglen);
            }
        }

        if (found_done) {
            return 0;
        }

        if (msg.msg_flags & MSG_TRUNC) {
```

```
            fprintf(stderr, "Message truncated\n");
            continue;
        }
        if (msglen) {
            fprintf(stderr, "!!!Remnant of size %d\n", msglen);
            exit(1);
        }
    }
}
```

7.15.3.21 int rtnl_dump_request (struct rtnl_handle * *rth*, int *type*, void * *req*, int *len*)

Definition at line 152 of file libnetlink.c.

References rtnl_handle::dump, rtnl_handle::fd, and rtnl_handle::seq.

```
                                                                    {
    struct nlmsghdr nlh;
    struct sockaddr_nl nladdr = { .nl_family = AF_NETLINK };
    struct iovec iov[2] = {
        { .iov_base = &nlh, .iov_len = sizeof(nlh) },
        { .iov_base = req, .iov_len = len }
    };
    struct msghdr msg = {
        .msg_name = &nladdr,
        .msg_namelen =  sizeof(nladdr),
        .msg_iov = iov,
        .msg_iovlen = 2,
    };

    nlh.nlmsg_len = NLMSG_LENGTH(len);
    nlh.nlmsg_type = type;
    nlh.nlmsg_flags = NLM_F_DUMP|NLM_F_REQUEST;
    nlh.nlmsg_pid = 0;
    nlh.nlmsg_seq = rth->dump = ++rth->seq;

    return sendmsg(rth->fd, &msg, 0);
}
```

7.15.3.22 int rtnl_from_file (FILE * , rtnl_filter_t *handler*, void * *jarg*)

Definition at line 472 of file libnetlink.c.

```
                            {
    int status;
    struct sockaddr_nl nladdr;
    char   buf[8192];
    struct nlmsghdr *h = (void *)buf;

    memset(&nladdr, 0, sizeof(nladdr));
    nladdr.nl_family = AF_NETLINK;
    nladdr.nl_pid = 0;
    nladdr.nl_groups = 0;

    while (1) {
        int err, len;
        int l;

        status = fread(&buf, 1, sizeof(*h), rtnl);

        if (status < 0) {
            if (errno == EINTR) {
                continue;
            }
            perror("rtnl_from_file: fread");
            return -1;
        }
        if (status == 0) {
            return 0;
        }

        len = h->nlmsg_len;
        l = len - sizeof(*h);

        if (l<0 || len>sizeof(buf)) {
            fprintf(stderr, "!!!malformed message: len=%d @%lu\n",
                    len, ftell(rtnl));
```

```
            return -1;
        }

        status = fread(NLMSG_DATA(h), 1, NLMSG_ALIGN(l), rtnl);

        if (status < 0) {
            perror("rtnl_from_file: fread");
            return -1;
        }
        if (status < l) {
            fprintf(stderr, "rtnl-from_file: truncated message\n");
            return -1;
        }

        err = handler(&nladdr, h, jarg);
        if (err < 0) {
            return err;
        }
    }
}
```

7.15.3.23 int rtnl_listen (struct rtnl_handle * , rtnl_filter_t *handler,* void * *jarg*)

Definition at line 395 of file libnetlink.c.

References rtnl_handle::fd.

```
                            {
    int status;
    struct nlmsghdr *h;
    struct sockaddr_nl nladdr;
    struct iovec iov;
    struct msghdr msg = {
        .msg_name = &nladdr,
        .msg_namelen = sizeof(nladdr),
        .msg_iov = &iov,
        .msg_iovlen = 1,
    };
    char   buf[8192];

    memset(&nladdr, 0, sizeof(nladdr));
    nladdr.nl_family = AF_NETLINK;
    nladdr.nl_pid = 0;
    nladdr.nl_groups = 0;

    iov.iov_base = buf;
    while (1) {
        iov.iov_len = sizeof(buf);
        status = recvmsg(rtnl->fd, &msg, 0);

        if (status < 0) {
            if (errno == EINTR || errno == EAGAIN) {
                continue;
            }
            fprintf(stderr, "netlink receive error %s (%d)\n",
                    strerror(errno), errno);
            if (errno == ENOBUFS) {
                continue;
            }
            return -1;
        }
        if (status == 0) {
            fprintf(stderr, "EOF on netlink\n");
            return -1;
        }
        if (msg.msg_namelen != sizeof(nladdr)) {
            fprintf(stderr, "Sender address length == %d\n", msg.msg_namelen);
            exit(1);
        }
        for (h = (struct nlmsghdr *)buf; status >= sizeof(*h); ) {
            int err;
            int len = h->nlmsg_len;
            int l = len - sizeof(*h);

            if (l<0 || len>status) {
                if (msg.msg_flags & MSG_TRUNC) {
                    fprintf(stderr, "Truncated message\n");
                    return -1;
                }
                fprintf(stderr, "!!!malformed message: len=%d\n", len);
                exit(1);
            }
```

```
            err = handler(&nladdr, h, jarg);
            if (err < 0) {
                return err;
            }

            status -= NLMSG_ALIGN(len);
            h = (struct nlmsghdr *)((char *)h + NLMSG_ALIGN(len));
        }
        if (msg.msg_flags & MSG_TRUNC) {
            fprintf(stderr, "Message truncated\n");
            continue;
        }
        if (status) {
            fprintf(stderr, "!!!Remnant of size %d\n", status);
            exit(1);
        }
    }
}
```

7.15.3.24 int rtnl_open (struct rtnl_handle * *rth*, unsigned *subscriptions*)

Definition at line 85 of file libnetlink.c.

References rtnl_open_byproto().

```
                                                                    {
    return rtnl_open_byproto(rth, subscriptions, NETLINK_ROUTE
      );
}
```

7.15.3.25 int rtnl_open_byproto (struct rtnl_handle * *rth*, unsigned *subscriptions*, int *protocol*)

Definition at line 37 of file libnetlink.c.

References rtnl_handle::fd, rtnl_handle::local, rcvbuf, and rtnl_handle::seq.

```
                                    {
    socklen_t addr_len;
    int sndbuf = 32768;

    memset(rth, 0, sizeof(*rth));

    rth->fd = socket(AF_NETLINK, SOCK_RAW, protocol);
    if (rth->fd < 0) {
        perror("Cannot open netlink socket");
        return -1;
    }

    if (setsockopt(rth->fd,SOL_SOCKET,SO_SNDBUF,&sndbuf,sizeof(sndbuf)) < 0)
      {
        perror("SO_SNDBUF");
        return -1;
    }

    if (setsockopt(rth->fd,SOL_SOCKET,SO_RCVBUF,&rcvbuf,sizeof(rcvbuf
      )) < 0) {
        perror("SO_RCVBUF");
        return -1;
    }

    memset(&rth->local, 0, sizeof(rth->local));
    rth->local.nl_family = AF_NETLINK;
    rth->local.nl_groups = subscriptions;

    if (bind(rth->fd, (struct sockaddr *)&rth->local, sizeof(rth->local
      )) < 0) {
        perror("Cannot bind netlink socket");
        return -1;
    }
    addr_len = sizeof(rth->local);
    if (getsockname(rth->fd, (struct sockaddr *)&rth->local, &addr_len)
      < 0) {
        perror("Cannot getsockname");
        return -1;
    }
```

```
    if (addr_len != sizeof(rth->local)) {
        fprintf(stderr, "Wrong address length %d\n", addr_len);
        return -1;
    }
    if (rth->local.nl_family != AF_NETLINK) {
        fprintf(stderr, "Wrong address family %d\n", rth->local.nl_family)
      ;
        return -1;
    }
    rth->seq = time(NULL);
    return 0;
}
```

7.15.3.26 int rtnl_send (struct rtnl_handle * *rth*, const void * *buf*, int)

Definition at line 113 of file libnetlink.c.

References rtnl_handle::fd.

```
                                                                    {
    return send(rth->fd, buf, len, 0);
}
```

7.15.3.27 int rtnl_send_check (struct rtnl_handle * *rth*, const void * *buf*, int)

Definition at line 117 of file libnetlink.c.

References rtnl_handle::fd.

```
                                                                        {
    struct nlmsghdr *h;
    int status;
    char resp[1024];

    status = send(rth->fd, buf, len, 0);
    if (status < 0) {
        return status;
    }

    /* Check for immediate errors */
    status = recv(rth->fd, resp, sizeof(resp), MSG_DONTWAIT|MSG_PEEK);
    if (status < 0) {
        if (errno == EAGAIN) {
            return 0;
        }
        return -1;
    }

    for (h = (struct nlmsghdr *)resp; NLMSG_OK(h, status);
         h = NLMSG_NEXT(h, status)) {
        if (h->nlmsg_type == NLMSG_ERROR) {
            struct nlmsgerr *err = (struct nlmsgerr *)NLMSG_DATA(h);
            if (h->nlmsg_len < NLMSG_LENGTH(sizeof(struct nlmsgerr))) {
                fprintf(stderr, "ERROR truncated\n");
            } else {
                errno = -err->error;
            }
            return -1;
        }
    }

    return 0;
}
```

7.15.3.28 int rtnl_talk (struct rtnl_handle * *rtnl*, struct nlmsghdr * *n*, pid_t *peer*, unsigned *groups*, struct nlmsghdr * *answer*)

Definition at line 275 of file libnetlink.c.

References rtnl_handle::fd, rtnl_handle::local, and rtnl_handle::seq.

```
                                                {
    int status;
    unsigned seq;
    struct nlmsghdr *h;
    struct sockaddr_nl nladdr;
    struct iovec iov = {
        .iov_base = (void *) n,
        .iov_len = n->nlmsg_len
    };
    struct msghdr msg = {
        .msg_name = &nladdr,
        .msg_namelen = sizeof(nladdr),
        .msg_iov = &iov,
        .msg_iovlen = 1,
    };
    char   buf[16384];

    memset(&nladdr, 0, sizeof(nladdr));
    nladdr.nl_family = AF_NETLINK;
    nladdr.nl_pid = peer;
    nladdr.nl_groups = groups;

    n->nlmsg_seq = seq = ++rtnl->seq;

    if (answer == NULL) {
        n->nlmsg_flags |= NLM_F_ACK;
    }

    status = sendmsg(rtnl->fd, &msg, 0);

    if (status < 0) {
        perror("Cannot talk to rtnetlink");
        return -1;
    }

    memset(buf,0,sizeof(buf));

    iov.iov_base = buf;

    while (1) {
        iov.iov_len = sizeof(buf);
        status = recvmsg(rtnl->fd, &msg, 0);

        if (status < 0) {
            if (errno == EINTR || errno == EAGAIN) {
                continue;
            }
            fprintf(stderr, "netlink receive error %s (%d)\n",
                    strerror(errno), errno);
            return -1;
        }
        if (status == 0) {
            fprintf(stderr, "EOF on netlink\n");
            return -1;
        }
        if (msg.msg_namelen != sizeof(nladdr)) {
            fprintf(stderr, "sender address length == %d\n", msg.msg_namelen);
            exit(1);
        }
        for (h = (struct nlmsghdr *)buf; status >= sizeof(*h); ) {
            int len = h->nlmsg_len;
            int l = len - sizeof(*h);

            if (l < 0 || len>status) {
                if (msg.msg_flags & MSG_TRUNC) {
                    fprintf(stderr, "Truncated message\n");
                    return -1;
                }
                fprintf(stderr, "!!!malformed message: len=%d\n", len);
                exit(1);
            }

            if (nladdr.nl_pid != peer ||
                    h->nlmsg_pid != rtnl->local.nl_pid ||
                    h->nlmsg_seq != seq) {
                /* Don't forget to skip that message. */
                status -= NLMSG_ALIGN(len);
                h = (struct nlmsghdr *)((char *)h + NLMSG_ALIGN(len));
                continue;
            }

            if (h->nlmsg_type == NLMSG_ERROR) {
                struct nlmsgerr *err = (struct nlmsgerr *)NLMSG_DATA(h);
                if (l < sizeof(struct nlmsgerr)) {
                    fprintf(stderr, "ERROR truncated\n");
                } else {
                    if (!err->error) {
```

```
                    if (answer) {
                        memcpy(answer, h, h->nlmsg_len);
                    }
                    return 0;
                }

                fprintf(stderr, "RTNETLINK answers: %s\n", strerror(-err->
    error));
                errno = -err->error;
            }
            return -1;
        }
        if (answer) {
            memcpy(answer, h, h->nlmsg_len);
            return 0;
        }

        fprintf(stderr, "Unexpected reply!!!\n");

        status -= NLMSG_ALIGN(len);
        h = (struct nlmsghdr *)((char *)h + NLMSG_ALIGN(len));
    }
    if (msg.msg_flags & MSG_TRUNC) {
        fprintf(stderr, "Message truncated\n");
        continue;
    }
    if (status) {
        fprintf(stderr, "!!!Remnant of size %d\n", status);
        exit(1);
    }
  }
}
```

7.15.3.29 int rtnl_wilddump_request (struct rtnl_handle * *rth*, int *fam*, int *type*)

Definition at line 89 of file libnetlink.c.

References rtnl_handle::dump, rtnl_handle::fd, and rtnl_handle::seq.

```
                                                                    {
    struct {
        struct nlmsghdr nlh;
        struct rtgenmsg g;
        __u16 align_rta;    /* attribute has to be 32bit aligned */
        struct rtattr ext_req;
        __u32 ext_filter_mask;
    } req;

    memset(&req, 0, sizeof(req));
    req.nlh.nlmsg_len = sizeof(req);
    req.nlh.nlmsg_type = type;
    req.nlh.nlmsg_flags = NLM_F_DUMP|NLM_F_REQUEST;
    req.nlh.nlmsg_pid = 0;
    req.nlh.nlmsg_seq = rth->dump = ++rth->seq;
    req.g.rtgen_family = family;

    req.ext_req.rta_type = IFLA_EXT_MASK;
    req.ext_req.rta_len = RTA_LENGTH(sizeof(__u32));
    req.ext_filter_mask = RTEXT_FILTER_VF;

    return send(rth->fd, (void *)&req, sizeof(req), 0);
}
```

7.15.4 Variable Documentation

7.15.4.1 int rcvbuf

Definition at line 28 of file libnetlink.c.

7.16 src/libnetlink/include/ll_map.h File Reference

Functions

- int ll_remember_index (const struct sockaddr_nl *who, struct nlmsghdr *n, void *arg)
- int ll_init_map (struct rtnl_handle *rth, int reinit)
- unsigned ll_name_to_index (const char *name)
- const char * ll_index_to_name (unsigned idx)
- const char * ll_idx_n2a (unsigned idx, char *buf)
- int ll_index_to_type (unsigned idx)
- unsigned ll_index_to_flags (unsigned idx)
- unsigned ll_index_to_addr (unsigned idx, unsigned char *addr, unsigned alen)

7.16.1 Function Documentation

7.16.1.1 const char* ll_idx_n2a (unsigned *idx*, char * *buf*)

Definition at line 99 of file ll_map.c.

References ll_cache::idx_next, ll_cache::index, and ll_cache::name.

Referenced by ll_index_to_name().

```
                                                {
    const struct ll_cache *im;

    if (idx == 0) {
        return "*";
    }

    for (im = idxhead(idx); im; im = im->idx_next)
        if (im->index == idx) {
            return im->name;
        }

    snprintf(buf, IFNAMSIZ, "if%d", idx);
    return buf;
}
```

7.16.1.2 unsigned ll_index_to_addr (unsigned *idx*, unsigned char * *addr*, unsigned *alen*)

Definition at line 149 of file ll_map.c.

References ll_cache::addr, ll_cache::alen, ll_cache::idx_next, and ll_cache::index.

Referenced by ifhwaddr().

```
                                                {
    const struct ll_cache *im;

    if (idx == 0) {
        return 0;
    }

    for (im = idxhead(idx); im; im = im->idx_next) {
        if (im->index == idx) {
            if (alen > sizeof(im->addr)) {
                alen = sizeof(im->addr);
            }
            if (alen > im->alen) {
                alen = im->alen;
            }
            memcpy(addr, im->addr, alen);
            return alen;
        }
    }
    return 0;
}
```

7.16.1.3 unsigned ll_index_to_flags (unsigned *idx*)

Definition at line 135 of file ll_map.c.

References ll_cache::flags, ll_cache::idx_next, and ll_cache::index.

Referenced by set_interface_flags().

```
                                        {
    const struct ll_cache *im;

    if (idx == 0) {
        return 0;
    }

    for (im = idxhead(idx); im; im = im->idx_next)
        if (im->index == idx) {
            return im->flags;
        }
    return 0;
}
```

7.16.1.4 const char* ll_index_to_name (unsigned *idx*)

Definition at line 116 of file ll_map.c.

References ll_idx_n2a().

```
                                        {
    static char nbuf[IFNAMSIZ];

    return ll_idx_n2a(idx, nbuf);
}
```

7.16.1.5 int ll_index_to_type (unsigned *idx*)

Definition at line 122 of file ll_map.c.

References ll_cache::idx_next, ll_cache::index, and ll_cache::type.

```
                                    {
    const struct ll_cache *im;

    if (idx == 0) {
        return -1;
    }
    for (im = idxhead(idx); im; im = im->idx_next)
        if (im->index == idx) {
            return im->type;
        }
    return -1;
}
```

7.16.1.6 int ll_init_map (struct rtnl_handle * *rth*, int *reinit*)

Definition at line 204 of file ll_map.c.

References ll_remember_index(), rtnl_dump_filter(), and rtnl_wilddump_request().

Referenced by get_iface_index().

```
                                                    {
    static int initialized;

    if (initialized && !reinit) {
        return 0;
    }
```

```
    if (rtnl_wilddump_request(rth, AF_UNSPEC, RTM_GETLINK)
       < 0) {
        perror("Cannot send dump request");
        exit(1);
    }

    if (rtnl_dump_filter(rth, ll_remember_index
      , NULL) < 0) {
        fprintf(stderr, "Dump terminated\n");
        exit(1);
    }

    initialized = 1;

    return 0;
}
```

7.16.1.7 unsigned ll_name_to_index (const char * *name*)

Definition at line 172 of file ll_map.c.

References ll_cache::idx_next, IDXMAP_SIZE, if_nametoindex(), ll_cache::index, and ll_cache::name.

Referenced by get_iface_index().

```
                                                {
    static char ncache[IFNAMSIZ];
    static int icache;
    struct ll_cache *im;
    int i;
    unsigned idx;

    if (name == NULL) {
        return 0;
    }

    if (icache && strcmp(name, ncache) == 0) {
        return icache;
    }

    for (i=0; i<IDXMAP_SIZE; i++) {
        for (im = idx_head[i]; im; im = im->idx_next) {
            if (strcmp(im->name, name) == 0) {
                icache = im->index;
                strcpy(ncache, name);
                return im->index;
            }
        }
    }

    idx = if_nametoindex(name);
    if (idx == 0) {
        sscanf(name, "if%u", &idx);
    }
    return idx;
}
```

7.16.1.8 int ll_remember_index (const struct sockaddr_nl * *who,* struct nlmsghdr * *n,* void * *arg*)

Definition at line 45 of file ll_map.c.

References ll_cache::addr, ll_cache::alen, ll_cache::flags, ll_cache::idx_next, IDXMAP_SIZE, IFLA_PAYLOAD, IFLA_RTA, ll_cache::index, malloc, ll_cache::name, parse_rtattr(), and ll_cache::type.

Referenced by ll_init_map().

```
                                                        {
    int h;
    struct ifinfomsg *ifi = NLMSG_DATA(n);
    struct ll_cache *im, **imp;
    struct rtattr *tb[IFLA_MAX+1];

    if (n->nlmsg_type != RTM_NEWLINK) {
        return 0;
    }
```

```
    if (n->nlmsg_len < NLMSG_LENGTH(sizeof(ifi))) {
        return -1;
    }

    memset(tb, 0, sizeof(tb));
    parse_rtattr(tb, IFLA_MAX, IFLA_RTA(ifi), IFLA_PAYLOAD
      (n));
    if (tb[IFLA_IFNAME] == NULL) {
        return 0;
    }

    h = ifi->ifi_index & (IDXMAP_SIZE - 1);
    for (imp = &idx_head[h]; (im=*imp)!=NULL; imp = &im->idx_next)
        if (im->index == ifi->ifi_index) {
            break;
        }

    if (im == NULL) {
        im = malloc(sizeof(*im));
        if (im == NULL) {
            return 0;
        }
        im->idx_next = *imp;
        im->index = ifi->ifi_index;
        *imp = im;
    }

    im->type = ifi->ifi_type;
    im->flags = ifi->ifi_flags;
    if (tb[IFLA_ADDRESS]) {
        int alen;
        im->alen = alen = RTA_PAYLOAD(tb[IFLA_ADDRESS]);
        if (alen > sizeof(im->addr)) {
            alen = sizeof(im->addr);
        }
        memcpy(im->addr, RTA_DATA(tb[IFLA_ADDRESS]), alen);
    } else {
        im->alen = 0;
        memset(im->addr, 0, sizeof(im->addr));
    }
    strcpy(im->name, RTA_DATA(tb[IFLA_IFNAME]));
    return 0;
}
```

7.17 src/libnetlink/include/rt_names.h File Reference

```
#include <asm/types.h>
```

Functions

- char * rtnl_rtprot_n2a (int id, char *buf, int len)
- char * rtnl_rtscope_n2a (int id, char *buf, int len)
- char * rtnl_rttable_n2a (__u32 id, char *buf, int len)
- char * rtnl_rtrealm_n2a (int id, char *buf, int len)
- char * rtnl_dsfield_n2a (int id, char *buf, int len)
- int rtnl_rtprot_a2n (__u32 *id, char *arg)
- int rtnl_rtscope_a2n (__u32 *id, char *arg)
- int rtnl_rttable_a2n (__u32 *id, char *arg)
- int rtnl_rtrealm_a2n (__u32 *id, char *arg)
- int rtnl_dsfield_a2n (__u32 *id, char *arg)
- int rtnl_group_a2n (int *id, char *arg)
- const char * inet_proto_n2a (int proto, char *buf, int len)
- int inet_proto_a2n (char *buf)
- const char * ll_type_n2a (int type, char *buf, int len)
- const char * ll_addr_n2a (unsigned char *addr, int alen, int type, char *buf, int blen)
- int ll_addr_a2n (char *lladdr, int len, char *arg)
- const char * ll_proto_n2a (unsigned short id, char *buf, int len)
- int ll_proto_a2n (unsigned short *id, char *buf)

7.17.1 Function Documentation

7.17.1.1 int inet_proto_a2n (char * *buf*)

Definition at line 45 of file inet_proto.c.

References get_u8().

```
                                {
    static char ncache[16];
    static int icache = -1;
    struct protoent *pe;

    if (icache>=0 && strcmp(ncache, buf) == 0) {
        return icache;
    }

    if (buf[0] >= '0' && buf[0] <= '9') {
        __u8 ret;
        if (get_u8(&ret, buf, 10)) {
            return -1;
        }
        return ret;
    }

    pe = getprotobyname(buf);
    if (pe) {
        icache = pe->p_proto;
        strncpy(ncache, pe->p_name, 16);
        return pe->p_proto;
    }
    return -1;
}
```

7.17.1.2 const char* inet_proto_n2a (int *proto,* char * *buf,* int *len*)

Definition at line 25 of file inet_proto.c.

```
                                                        {
    static char ncache[16];
    static int icache = -1;
    struct protoent *pe;

    if (proto == icache) {
        return ncache;
    }

    pe = getprotobynumber(proto);
    if (pe) {
        icache = proto;
        strncpy(ncache, pe->p_name, 16);
        strncpy(buf, pe->p_name, len);
        return buf;
    }
    snprintf(buf, len, "ipproto-%d", proto);
    return buf;
}
```

7.17.1.3 int ll_addr_a2n (char * *lladdr,* int *len,* char * *arg*)

Definition at line 59 of file ll_addr.c.

References inet_prefix::data, and get_addr_1().

```
                                                      {
    if (strchr(arg, '.')) {
        inet_prefix pfx;
        if (get_addr_1(&pfx, arg, AF_INET)) {
            fprintf(stderr, "\"%s\" is invalid lladdr.\n", arg);
            return -1;
        }
        if (len < 4) {
            return -1;
```

```
        }
        memcpy(lladdr, pfx.data, 4);
        return 4;
    } else {
        int i;

        for (i=0; i<len; i++) {
            int temp;
            char *cp = strchr(arg, ':');
            if (cp) {
                *cp = 0;
                cp++;
            }
            if (sscanf(arg, "%x", &temp) != 1) {
                fprintf(stderr, "\"%s\" is invalid lladdr.\n", arg);
                return -1;
            }
            if (temp < 0 || temp > 255) {
                fprintf(stderr, "\"%s\" is invalid lladdr.\n", arg);
                return -1;
            }
            lladdr[i] = temp;
            if (!cp) {
                break;
            }
            arg = cp;
        }
        return i+1;
    }
}
```

7.17.1.4 const char* ll_addr_n2a (unsigned char * *addr*, int *alen*, int *type*, char * *buf*, int *blen*)

Definition at line 32 of file ll_addr.c.

```
          {
    int i;
    int l;

    if (alen == 4 &&
            (type == ARPHRD_TUNNEL || type == ARPHRD_SIT || type ==
      ARPHRD_IPGRE)) {
        return inet_ntop(AF_INET, addr, buf, blen);
    }
    if (alen == 16 && type == ARPHRD_TUNNEL6) {
        return inet_ntop(AF_INET6, addr, buf, blen);
    }
    l = 0;
    for (i=0; i<alen; i++) {
        if (i==0) {
            snprintf(buf+l, blen, "%02x", addr[i]);
            blen -= 2;
            l += 2;
        } else {
            snprintf(buf+l, blen, ":%02x", addr[i]);
            blen -= 3;
            l += 3;
        }
    }
    return buf;
}
```

7.17.1.5 int ll_proto_a2n (unsigned short * *id*, char * *buf*)

Definition at line 103 of file ll_proto.c.

References get_u16(), and name.

```
                                                {
    int i;
    for (i=0; i<sizeof(llproto_names)/sizeof(llproto_names[0]); i++) {
        if (strcasecmp(llproto_names[i].name, buf) == 0) {
            *id = htons(llproto_names[i].id);
            return 0;
        }
    }
```

```
    if (get_u16(id, buf, 0)) {
        return -1;
    }
    *id = htons(*id);
    return 0;
}
```

7.17.1.6 const char* ll_proto_n2a (unsigned short *id*, char * *buf*, int *len*)

Definition at line 89 of file ll_proto.c.

```
{
    int i;

    id = ntohs(id);

    for (i=0; i<sizeof(llproto_names)/sizeof(llproto_names[0]); i++) {
        if (llproto_names[i].id == id) {
            return llproto_names[i].name;
        }
    }
    snprintf(buf, len, "[%d]", id);
    return buf;
}
```

7.17.1.7 const char* ll_type_n2a (int *type*, char * *buf*, int *len*)

Definition at line 30 of file ll_types.c.

References __PF, and name.

```
{
#define __PF(f,n) { ARPHRD_##f, #n },
    static const struct {
        int type;
        const char *name;
    } arphrd_names[] = {
        { 0, "generic" },
        __PF(ETHER,ether)
        __PF(EETHER,eether)
        __PF(AX25,ax25)
        __PF(PRONET,pronet)
        __PF(CHAOS,chaos)
        __PF(IEEE802,ieee802)
        __PF(ARCNET,arcnet)
        __PF(APPLETLK,atalk)
        __PF(DLCI,dlci)
        __PF(ATM,atm)
        __PF(METRICOM,metricom)
        __PF(IEEE1394,ieee1394)
        __PF(INFINIBAND,infiniband)
        __PF(SLIP,slip)
        __PF(CSLIP,cslip)
        __PF(SLIP6,slip6)
        __PF(CSLIP6,cslip6)
        __PF(RSRVD,rsrvd)
        __PF(ADAPT,adapt)
        __PF(ROSE,rose)
        __PF(X25,x25)
        __PF(HWX25,hwx25)
        __PF(CAN,can)
        __PF(PPP,ppp)
        __PF(HDLC,hdlc)
        __PF(LAPB,lapb)
        __PF(DDCMP,ddcmp)
        __PF(RAWHDLC,rawhdlc)
        __PF(TUNNEL,ipip)
        __PF(TUNNEL6,tunnel6)
        __PF(FRAD,frad)
        __PF(SKIP,skip)
        __PF(LOOPBACK,loopback)
        __PF(LOCALTLK,ltalk)
        __PF(FDDI,fddi)
        __PF(BIF,bif)
        __PF(SIT,sit)
        __PF(IPDDP,ip/ddp)
        __PF(IPGRE,gre)
```

```
        __PF(PIMREG,pimreg)
        __PF(HIPPI,hippi)
        __PF(ASH,ash)
        __PF(ECONET,econet)
        __PF(IRDA,irda)
        __PF(FCPP,fcpp)
        __PF(FCAL,fcal)
        __PF(FCPL,fcpl)
        __PF(FCFABRIC,fcfb0)
        __PF(FCFABRIC+1,fcfb1)
        __PF(FCFABRIC+2,fcfb2)
        __PF(FCFABRIC+3,fcfb3)
        __PF(FCFABRIC+4,fcfb4)
        __PF(FCFABRIC+5,fcfb5)
        __PF(FCFABRIC+6,fcfb6)
        __PF(FCFABRIC+7,fcfb7)
        __PF(FCFABRIC+8,fcfb8)
        __PF(FCFABRIC+9,fcfb9)
        __PF(FCFABRIC+10,fcfb10)
        __PF(FCFABRIC+11,fcfb11)
        __PF(FCFABRIC+12,fcfb12)
        __PF(IEEE802_TR,tr)
        __PF(IEEE80211,ieee802.11)
        __PF(IEEE80211_PRISM,ieee802.11/prism)
        __PF(IEEE80211_RADIOTAP,ieee802.11/radiotap)
        __PF(IEEE802154, ieee802.15.4)
        __PF(PHONET, phonet)
        __PF(PHONET_PIPE, phonet_pipe)
        __PF(CAIF, caif)

        __PF(NONE, none)
        __PF(VOID,void)
    };
#undef __PF

    int i;
    for (i=0; i<sizeof(arphrd_names)/sizeof(arphrd_names[0]); i++) {
        if (arphrd_names[i].type == type) {
            return arphrd_names[i].name;
        }
    }
    snprintf(buf, len, "[%d]", type);
    return buf;
}
```

7.17.1.8 int rtnl_dsfield_a2n (__u32 * *id*, char * *arg*)

Definition at line 436 of file rt_names.c.

```
                                        {
    static char *cache = NULL;
    static unsigned long res;
    char *end;
    int i;

    if (cache && strcmp(cache, arg) == 0) {
        *id = res;
        return 0;
    }

    if (!rtnl_rtdsfield_init) {
        rtnl_rtdsfield_initialize();
    }

    for (i=0; i<256; i++) {
        if (rtnl_rtdsfield_tab[i] &&
                strcmp(rtnl_rtdsfield_tab[i], arg) == 0) {
            cache = rtnl_rtdsfield_tab[i];
            res = i;
            *id = res;
            return 0;
        }
    }

    res = strtoul(arg, &end, 16);
    if (!end || end == arg || *end || res > 255) {
        return -1;
    }
    *id = res;
    return 0;
}
```

7.17.1.9 char* rtnl_dsfield_n2a (int *id*, char * *buf*, int *len*)

Definition at line 418 of file rt_names.c.

References id.

```
                                                    {
    if (id<0 || id>=256) {
        snprintf(buf, len, "%d", id);
        return buf;
    }
    if (!rtnl_rtdsfield_tab[id]) {
        if (!rtnl_rtdsfield_init) {
            rtnl_rtdsfield_initialize();
        }
    }
    if (rtnl_rtdsfield_tab[id]) {
        return rtnl_rtdsfield_tab[id];
    }
    snprintf(buf, len, "0x%02x", id);
    return buf;
}
```

7.17.1.10 int rtnl_group_a2n (int * *id*, char * *arg*)

Definition at line 484 of file rt_names.c.

References rtnl_hash_entry::id, rtnl_hash_entry::name, and rtnl_hash_entry::next.

```
                                        {
    static char *cache = NULL;
    static unsigned long res;
    struct rtnl_hash_entry *entry;
    char *end;
    int i;

    if (cache && strcmp(cache, arg) == 0) {
        *id = res;
        return 0;
    }

    if (!rtnl_group_init) {
        rtnl_group_initialize();
    }

    for (i=0; i<256; i++) {
        entry = rtnl_group_hash[i];
        while (entry && strcmp(entry->name, arg)) {
            entry = entry->next;
        }
        if (entry) {
            cache = entry->name;
            res = entry->id;
            *id = res;
            return 0;
        }
    }

    i = strtol(arg, &end, 0);
    if (!end || end == arg || *end || i < 0) {
        return -1;
    }
    *id = i;
    return 0;
}
```

7.17.1.11 int rtnl_rtprot_a2n (__u32 * *id*, char * *arg*)

Definition at line 162 of file rt_names.c.

```
                                          {
    static char *cache = NULL;
    static unsigned long res;
    char *end;
```

```
    int i;

    if (cache && strcmp(cache, arg) == 0) {
        *id = res;
        return 0;
    }

    if (!rtnl_rtprot_init) {
        rtnl_rtprot_initialize();
    }

    for (i=0; i<256; i++) {
        if (rtnl_rtprot_tab[i] &&
                strcmp(rtnl_rtprot_tab[i], arg) == 0) {
            cache = rtnl_rtprot_tab[i];
            res = i;
            *id = res;
            return 0;
        }
    }

    res = strtoul(arg, &end, 0);
    if (!end || end == arg || *end || res > 255) {
        return -1;
    }
    *id = res;
    return 0;
}
```

7.17.1.12 char* rtnl_rtprot_n2a (int *id*, char * *buf*, int *len*)

Definition at line 145 of file rt_names.c.

References id.

```
                                                    {
    if (id<0 || id>=256) {
        snprintf(buf, len, "%d", id);
        return buf;
    }
    if (!rtnl_rtprot_tab[id]) {
        if (!rtnl_rtprot_init) {
            rtnl_rtprot_initialize();
        }
    }
    if (rtnl_rtprot_tab[id]) {
        return rtnl_rtprot_tab[id];
    }
    snprintf(buf, len, "%d", id);
    return buf;
}
```

7.17.1.13 int rtnl_rtrealm_a2n (__u32 * *id*, char * *arg*)

Definition at line 295 of file rt_names.c.

```
                                                {
    static char *cache = NULL;
    static unsigned long res;
    char *end;
    int i;

    if (cache && strcmp(cache, arg) == 0) {
        *id = res;
        return 0;
    }

    if (!rtnl_rtrealm_init) {
        rtnl_rtrealm_initialize();
    }

    for (i=0; i<256; i++) {
        if (rtnl_rtrealm_tab[i] &&
                strcmp(rtnl_rtrealm_tab[i], arg) == 0) {
            cache = rtnl_rtrealm_tab[i];
            res = i;
            *id = res;
```

```
            return 0;
        }
    }

    res = strtoul(arg, &end, 0);
    if (!end || end == arg || *end || res > 255) {
        return -1;
    }
    *id = res;
    return 0;
}
```

7.17.1.14 char* rtnl_rtrealm_n2a (int *id*, char * *buf*, int *len*)

Definition at line 277 of file rt_names.c.

References id.

```
                                                    {
    if (id<0 || id>=256) {
        snprintf(buf, len, "%d", id);
        return buf;
    }
    if (!rtnl_rtrealm_tab[id]) {
        if (!rtnl_rtrealm_init) {
            rtnl_rtrealm_initialize();
        }
    }
    if (rtnl_rtrealm_tab[id]) {
        return rtnl_rtrealm_tab[id];
    }
    snprintf(buf, len, "%d", id);
    return buf;
}
```

7.17.1.15 int rtnl_rtscope_a2n (__u32 * *id*, char * *arg*)

Definition at line 230 of file rt_names.c.

```
                                          {
    static char *cache = NULL;
    static unsigned long res;
    char *end;
    int i;

    if (cache && strcmp(cache, arg) == 0) {
        *id = res;
        return 0;
    }

    if (!rtnl_rtscope_init) {
        rtnl_rtscope_initialize();
    }

    for (i=0; i<256; i++) {
        if (rtnl_rtscope_tab[i] &&
                strcmp(rtnl_rtscope_tab[i], arg) == 0) {
            cache = rtnl_rtscope_tab[i];
            res = i;
            *id = res;
            return 0;
        }
    }

    res = strtoul(arg, &end, 0);
    if (!end || end == arg || *end || res > 255) {
        return -1;
    }
    *id = res;
    return 0;
}
```

7.17.1.16 char* rtnl_rtscope_n2a (int *id*, char * *buf*, int *len*)

Definition at line 213 of file rt_names.c.

References id.

```
{
    if (id<0 || id>=256) {
        snprintf(buf, len, "%d", id);
        return buf;
    }
    if (!rtnl_rtscope_tab[id]) {
        if (!rtnl_rtscope_init) {
            rtnl_rtscope_initialize();
        }
    }
    if (rtnl_rtscope_tab[id]) {
        return rtnl_rtscope_tab[id];
    }
    snprintf(buf, len, "%d", id);
    return buf;
}
```

7.17.1.17 int rtnl_rttable_a2n (__u32 * *id*, char * *arg*)

Definition at line 368 of file rt_names.c.

References rtnl_hash_entry::id, rtnl_hash_entry::name, and rtnl_hash_entry::next.

```
{
    static char *cache = NULL;
    static unsigned long res;
    struct rtnl_hash_entry *entry;
    char *end;
    __u32 i;

    if (cache && strcmp(cache, arg) == 0) {
        *id = res;
        return 0;
    }

    if (!rtnl_rttable_init) {
        rtnl_rttable_initialize();
    }

    for (i=0; i<256; i++) {
        entry = rtnl_rttable_hash[i];
        while (entry && strcmp(entry->name, arg)) {
            entry = entry->next;
        }
        if (entry) {
            cache = entry->name;
            res = entry->id;
            *id = res;
            return 0;
        }
    }

    i = strtoul(arg, &end, 0);
    if (!end || end == arg || *end || i > RT_TABLE_MAX) {
        return -1;
    }
    *id = i;
    return 0;
}
```

7.17.1.18 char* rtnl_rttable_n2a (__u32 *id*, char * *buf*, int *len*)

Definition at line 347 of file rt_names.c.

References rtnl_hash_entry::id, rtnl_hash_entry::name, and rtnl_hash_entry::next.

```
{
    struct rtnl_hash_entry *entry;
```

```
    if (id > RT_TABLE_MAX) {
        snprintf(buf, len, "%u", id);
        return buf;
    }
    if (!rtnl_rttable_init) {
        rtnl_rttable_initialize();
    }
    entry = rtnl_rttable_hash[id & 255];
    while (entry && entry->id != id) {
        entry = entry->next;
    }
    if (entry) {
        return entry->name;
    }
    snprintf(buf, len, "%u", id);
    return buf;
}
```

7.18 src/libnetlink/include/rtm_map.h File Reference

Functions

- char * rtnl_rtntype_n2a (int id, char *buf, int len)
- int rtnl_rtntype_a2n (int *id, char *arg)
- int get_rt_realms (__u32 *realms, char *arg)

7.18.1 Function Documentation

7.18.1.1 int get_rt_realms (__u32 * *realms*, char * *arg*)

7.18.1.2 int rtnl_rtntype_a2n (int * *id*, char * *arg*)

7.18.1.3 char* rtnl_rtntype_n2a (int *id*, char * *buf*, int *len*)

7.19 src/libnetlink/include/utils.h File Reference

```
#include <asm/types.h>
#include <resolv.h>
#include <stdlib.h>
#include "libnetlink.h"
#include "ll_map.h"
#include "rtm_map.h"
```

Data Structures

- struct inet_prefix
- struct dn_naddr
- struct ipx_addr

Macros

- #define IPPROTO_ESP 50
- #define IPPROTO_AH 51
- #define IPPROTO_COMP 108
- #define IPSEC_PROTO_ANY 255
- #define SPRINT_BSIZE 64

- #define SPRINT_BUF(x) char x[SPRINT_BSIZE]
- #define NEXT_ARG() do { argv++; if (--argc <= 0) incomplete_command(); } while(0)
- #define NEXT_ARG_OK() (argc - 1 > 0)
- #define PREV_ARG() do { argv--; argc++; } while(0)
- #define PREFIXLEN_SPECIFIED 1
- #define DN_MAXADDL 20
- #define AF_DECnet 12
- #define IPX_NODE_LEN 6
- #define get_byte get_u8
- #define get_ushort get_u16
- #define get_short get_s16
- #define ARRAY_SIZE(x) (sizeof(x) / sizeof((x)[0]))

Functions

- void incomplete_command (void) __attribute__((noreturn))
- __u32 get_addr32 (const char *name)
- int get_addr_1 (inet_prefix *dst, const char *arg, int family)
- int get_prefix_1 (inet_prefix *dst, char *arg, int family)
- int get_addr (inet_prefix *dst, const char *arg, int family)
- int get_prefix (inet_prefix *dst, char *arg, int family)
- int mask2bits (__u32 netmask)
- int get_integer (int *val, const char *arg, int base)
- int get_unsigned (unsigned *val, const char *arg, int base)
- int get_time_rtt (unsigned *val, const char *arg, int *raw)
- int get_u64 (__u64 *val, const char *arg, int base)
- int get_u32 (__u32 *val, const char *arg, int base)
- int get_s32 (__s32 *val, const char *arg, int base)
- int get_u16 (__u16 *val, const char *arg, int base)
- int get_s16 (__s16 *val, const char *arg, int base)
- int get_u8 (__u8 *val, const char *arg, int base)
- int get_s8 (__s8 *val, const char *arg, int base)
- char * hexstring_n2a (const __u8 *str, int len, char *buf, int blen)
- __u8 * hexstring_a2n (const char *str, __u8 *buf, int blen)
- const char * format_host (int af, int len, const void *addr, char *buf, int buflen)
- const char * rt_addr_n2a (int af, int len, const void *addr, char *buf, int buflen)
- void missarg (const char *) __attribute__((noreturn))
- void invarg (const char *, const char *) __attribute__((noreturn))
- void duparg (const char *, const char *) __attribute__((noreturn))
- void duparg2 (const char *, const char *) __attribute__((noreturn))
- int matches (const char *arg, const char *pattern)
- int inet_addr_match (const inet_prefix *a, const inet_prefix *b, int bits)
- const char * dnet_ntop (int af, const void *addr, char *str, size_t len)
- int dnet_pton (int af, const char *src, void *addr)
- const char * ipx_ntop (int af, const void *addr, char *str, size_t len)
- int ipx_pton (int af, const char *src, void *addr)
- int __get_hz (void)
- int __get_user_hz (void)
- int print_timestamp (FILE *fp)
- ssize_t getcmdline (char **line, size_t *len, FILE *in)
- int makeargs (char *line, char *argv[], int maxargs)
- int iplink_parse (int argc, char **argv, struct iplink_req *req, char **name, char **type, char **link, char **dev, int *group)

Variables

- int preferred_family
- int show_stats
- int show_details
- int show_raw
- int resolve_hosts
- int oneline
- int timestamp
- char * _SL_
- int max_flush_loops
- int __iproute2_hz_internal
- int __iproute2_user_hz_internal
- int cmdlineno

7.19.1 Macro Definition Documentation

7.19.1.1 #define AF_DECnet 12

Definition at line 56 of file utils.h.

Referenced by dnet_ntop(), dnet_pton(), format_host(), get_addr_1(), get_prefix_1(), and rt_addr_n2a().

7.19.1.2 #define ARRAY_SIZE(*x*) (sizeof(x) / sizeof((x)[0]))

Definition at line 144 of file utils.h.

7.19.1.3 #define DN_MAXADDL 20

Definition at line 54 of file utils.h.

7.19.1.4 #define get_byte get_u8

Definition at line 81 of file utils.h.

7.19.1.5 #define get_short get_s16

Definition at line 83 of file utils.h.

7.19.1.6 #define get_ushort get_u16

Definition at line 82 of file utils.h.

7.19.1.7 #define IPPROTO_AH 51

Definition at line 26 of file utils.h.

7.19.1.8 #define IPPROTO_COMP 108

Definition at line 29 of file utils.h.

7.19.1.9 #define IPPROTO_ESP 50

Definition at line 23 of file utils.h.

7.19.1.10 #define IPSEC_PROTO_ANY 255

Definition at line 32 of file utils.h.

7.19.1.11 #define IPX_NODE_LEN 6

Definition at line 64 of file utils.h.

7.19.1.12 #define NEXT_ARG( ) do { argv++; if (--argc <= 0) incomplete_command(); } while(0)

Definition at line 40 of file utils.h.

7.19.1.13 #define NEXT_ARG_OK( ) (argc - 1 > 0)

Definition at line 41 of file utils.h.

7.19.1.14 #define PREFIXLEN_SPECIFIED 1

Definition at line 52 of file utils.h.

Referenced by get_prefix_1().

7.19.1.15 #define PREV_ARG( ) do { argv--; argc++; } while(0)

Definition at line 42 of file utils.h.

7.19.1.16 #define SPRINT_BSIZE 64

Definition at line 35 of file utils.h.

7.19.1.17 #define SPRINT_BUF(*x*) char x[SPRINT_BSIZE]

Definition at line 36 of file utils.h.

7.19.2 Function Documentation

7.19.2.1 int __get_hz (void)

Definition at line 502 of file utils.c.

References name.

```
                {
char name[1024];
int hz = 0;
FILE *fp;

if (getenv("HZ")) {
    return atoi(getenv("HZ")) ? : HZ;
}
```

```
    if (getenv("PROC_NET_PSCHED")) {
        snprintf(name, sizeof(name)-1, "%s", getenv("PROC_NET_PSCHED"));
    } else
        if (getenv("PROC_ROOT")) {
            snprintf(name, sizeof(name)-1, "%s/net/psched", getenv("PROC_ROOT")
      );
        } else {
            strcpy(name, "/proc/net/psched");
        }
    fp = fopen(name, "r");

    if (fp) {
        unsigned nom, denom;
        if (fscanf(fp, "%*08x%*08x%08x%08x", &nom, &denom) == 2)
            if (nom == 1000000) {
                hz = denom;
            }
        fclose(fp);
    }
    if (hz) {
        return hz;
    }
    return HZ;
}
```

7.19.2.2 int __get_user_hz (void)

Definition at line 537 of file utils.c.

```
                        {
    return sysconf(_SC_CLK_TCK);
}
```

7.19.2.3 const char* dnet_ntop (int *af*, const void * *addr*, char * *str*, size_t *len*)

Definition at line 97 of file dnet_ntop.c.

References AF_DECnet.

Referenced by rt_addr_n2a().

```
                                                                        {
    switch(af) {
        case AF_DECnet:
            errno = 0;
            return dnet_ntop1((struct dn_naddr *)addr, str, len);
        default
                :
            errno = EAFNOSUPPORT;
    }

    return NULL;
}
```

7.19.2.4 int dnet_pton (int *af*, const char * *src*, void * *addr*)

Definition at line 59 of file dnet_pton.c.

References AF_DECnet.

Referenced by get_addr_1().

```
                                                {
    int err;

    switch (af) {
        case AF_DECnet:
            errno = 0;
            err = dnet_pton1(src, (struct dn_naddr *)addr);
            break;
        default
```

```
                :
            errno = EAFNOSUPPORT;
            err = -1;
    }

    return err;
}
```

7.19.2.5 void duparg (const char * , const char *)

Definition at line 453 of file utils.c.

```
                                                {
    fprintf(stderr, "Error: duplicate \"%s\": \"%s\" is the second value.\n",
      key, arg);
    exit(-1);
}
```

7.19.2.6 void duparg2 (const char * , const char *)

Definition at line 458 of file utils.c.

```
                                                    {
    fprintf(stderr, "Error: either \"%s\" is duplicate, or \"%s\" is a garbage.
      \n", key, arg);
    exit(-1);
}
```

7.19.2.7 const char* format_host (int *af,* int *len,* const void * *addr,* char * *buf,* int *buflen*)

Definition at line 616 of file utils.c.

References AF_DECnet, resolve_hosts, and rt_addr_n2a().

```
                                                    {
#ifdef RESOLVE_HOSTNAMES
    if (resolve_hosts) {
        const char *n;

        if (len <= 0) {
            switch (af) {
                case AF_INET:
                    len = 4;
                    break;
                case AF_INET6:
                    len = 16;
                    break;
                case AF_IPX:
                    len = 10;
                    break;
#ifdef AF_DECnet
                    /* I see no reasons why gethostbyname
                       may not work for DECnet */
                case AF_DECnet:
                    len = 2;
                    break;
#endif
                default
                        :
                    ;
            }
        }
        if (len > 0 &&
                (n = resolve_address(addr, len, af)) != NULL) {
            return n;
        }
    }
#endif
    return rt_addr_n2a(af, len, addr, buf, buflen);
}
```

7.19.2.8 int get_addr (inet_prefix * *dst*, const char * *arg*, int *family*)

Definition at line 405 of file utils.c.

References get_addr_1().

```
                                                                    {
    if (family == AF_PACKET) {
        fprintf(stderr, "Error: \"%s\" may be inet address, but it is not
       allowed in this context.\n", arg);
        exit(1);
    }
    if (get_addr_1(dst, arg, family)) {
        fprintf(stderr, "Error: an inet address is expected rather than \"%s\".
      \n", arg);
        exit(1);
    }
    return 0;
}
```

7.19.2.9 __u32 get_addr32 (const char * *name*)

Definition at line 429 of file utils.c.

References inet_prefix::data, and get_addr_1().

```
                                    {
    inet_prefix addr;
    if (get_addr_1(&addr, name, AF_INET)) {
        fprintf(stderr, "Error: an IP address is expected rather than \"%s\"\n"
      , name);
        exit(1);
    }
    return addr.data[0];
}
```

7.19.2.10 int get_addr_1 (inet_prefix * *dst*, const char * *arg*, int *family*)

Definition at line 296 of file utils.c.

References dn_naddr::a_addr, AF_DECnet, inet_prefix::bitlen, inet_prefix::bytelen, inet_prefix::data, dnet_pton(), and inet_prefix::family.

Referenced by get_addr(), get_addr32(), get_prefix_1(), and ll_addr_a2n().

```
                                                                    {
    memset(addr, 0, sizeof(*addr));

    if (strcmp(name, "default") == 0 ||
            strcmp(name, "all") == 0 ||
            strcmp(name, "any") == 0) {
        if (family == AF_DECnet) {
            return -1;
        }
        addr->family = family;
        addr->bytelen = (family == AF_INET6 ? 16 : 4);
        addr->bitlen = -1;
        return 0;
    }

    if (strchr(name, ':')) {
        addr->family = AF_INET6;
        if (family != AF_UNSPEC && family != AF_INET6) {
            return -1;
        }
        if (inet_pton(AF_INET6, name, addr->data) <= 0) {
            return -1;
        }
        addr->bytelen = 16;
        addr->bitlen = -1;
        return 0;
    }
```

```
    if (family == AF_DECnet) {
        struct dn_naddr dna;
        addr->family = AF_DECnet;
        if (dnet_pton(AF_DECnet, name, &dna) <= 0) {
            return -1;
        }
        memcpy(addr->data, dna.a_addr, 2);
        addr->bytelen = 2;
        addr->bitlen = -1;
        return 0;
    }

    addr->family = AF_INET;
    if (family != AF_UNSPEC && family != AF_INET) {
        return -1;
    }

    if (get_addr_ipv4((__u8 *)addr->data, name) <= 0) {
        return -1;
    }

    addr->bytelen = 4;
    addr->bitlen = -1;
    return 0;
}
```

7.19.2.11 int get_integer (int * *val*, const char * *arg*, int *base*)

Definition at line 33 of file utils.c.

```
{
    long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtol(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res > INT_MAX || res < INT_MIN) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.19.2.12 int get_prefix (inet_prefix * *dst*, char * *arg*, int *family*)

Definition at line 417 of file utils.c.

References get_prefix_1().

Referenced by set_interface_ipaddr().

```
{
    if (family == AF_PACKET) {
        fprintf(stderr, "Error: \"%s\" may be inet prefix, but it is not
       allowed in this context.\n", arg);
        exit(1);
    }
    if (get_prefix_1(dst, arg, family)) {
        fprintf(stderr, "Error: an inet prefix is expected rather than \"%s\".
      \n", arg);
        exit(1);
    }
    return 0;
}
```

7.19.2.13 int get_prefix_1 (inet_prefix * *dst*, char * *arg*, int *family*)

Definition at line 350 of file utils.c.

References AF_DECnet, inet_prefix::bitlen, inet_prefix::bytelen, inet_prefix::family, inet_prefix::flags, get_addr_1(), and PREFIXLEN_SPECIFIED.

Referenced by get_prefix().

```
                                                                {
    int err;
    unsigned plen;
    char *slash;

    memset(dst, 0, sizeof(*dst));

    if (strcmp(arg, "default") == 0 ||
            strcmp(arg, "any") == 0 ||
            strcmp(arg, "all") == 0) {
        if (family == AF_DECnet) {
            return -1;
        }
        dst->family = family;
        dst->bytelen = 0;
        dst->bitlen = 0;
        return 0;
    }

    slash = strchr(arg, '/');
    if (slash) {
        *slash = 0;
    }

    err = get_addr_1(dst, arg, family);
    if (err == 0) {
        switch(dst->family) {
            case AF_INET6:
                dst->bitlen = 128;
                break;
            case AF_DECnet:
                dst->bitlen = 16;
                break;
            default
                    :
            case AF_INET:
                dst->bitlen = 32;
        }
        if (slash) {
            if (get_netmask(&plen, slash+1, 0)
                    || plen > dst->bitlen) {
                err = -1;
                goto done;
            }
            dst->flags |= PREFIXLEN_SPECIFIED;
            dst->bitlen = plen;
        }
    }
done:
    if (slash) {
        *slash = '/';
    }
    return err;
}
```

7.19.2.14 int get_s16 (__s16 * *val*, const char * *arg*, int *base*)

Definition at line 232 of file utils.c.

```
                                                    {
    long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtol(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res > 0x7FFF || res < -0x8000) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.19.2.15 int get_s32 (__s32 * *val*, const char * *arg*, int *base*)

Definition at line 213 of file utils.c.

```
                                                    {
    long res;
    char *ptr;

    errno = 0;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtol(arg, &ptr, base);
    if (ptr == arg || *ptr ||
            ((res ==  LONG_MIN || res == LONG_MAX) && errno == ERANGE) ||
            res > INT32_MAX || res < INT32_MIN) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.19.2.16 int get_s8 (__s8 * *val*, const char * *arg*, int *base*)

Definition at line 247 of file utils.c.

```
                                                   {
    long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtol(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res > 0x7F || res < -0x80) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.19.2.17 int get_time_rtt (unsigned * *val*, const char * *arg*, int * *raw*)

Definition at line 106 of file utils.c.

```
                                                                {
    double t;
    unsigned long res;
    char *p;

    if (strchr(arg,'.') != NULL) {
        t = strtod(arg,&p);
        if (t < 0.0) {
            return -1;
        }
    } else {
        res = strtoul(arg, &p, 0);
        if (res > UINT_MAX) {
            return -1;
        }
        t = (double)res;
    }
    if (p == arg) {
        return -1;
    }
    *raw = 1;

    if (*p) {
        *raw = 0;
        if (strcasecmp(p, "s") == 0 || strcasecmp(p, "sec")==0 ||
                strcasecmp(p, "secs")==0) {
            t *= 1000;
        } else
            if (strcasecmp(p, "ms") == 0 || strcasecmp(p, "msec")==0 ||
```

```
                strcasecmp(p, "msecs") == 0) {
            t *= 1.0;    /* allow suffix, do nothing */
        } else {
            return -1;
        }
    }

    /* emulate ceil() without having to bring-in -lm and always be >= 1 */

    *val = t;
    if (*val < t) {
        *val += 1;
    }

    return 0;

}
```

7.19.2.18 int get_u16 (__u16 * *val*, const char * *arg*, int *base*)

Definition at line 183 of file utils.c.

Referenced by ll_proto_a2n().

```
                                                {
    unsigned long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtoul(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res > 0xFFFF) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.19.2.19 int get_u32 (__u32 * *val*, const char * *arg*, int *base*)

Definition at line 168 of file utils.c.

```
                                                {
    unsigned long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtoul(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res > 0xFFFFFFFFUL) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.19.2.20 int get_u64 (__u64 * *val*, const char * *arg*, int *base*)

Definition at line 153 of file utils.c.

```
                                                {
    unsigned long long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtoull(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res == 0xFFFFFFFFULL) {
```

```
        return -1;
    }
    *val = res;
    return 0;
}
```

7.19.2.21 int get_u8 (__u8 * *val*, const char * *arg*, int *base*)

Definition at line 198 of file utils.c.

Referenced by inet_proto_a2n().

```
{
    unsigned long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtoul(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res > 0xFF) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.19.2.22 int get_unsigned (unsigned * *val*, const char * *arg*, int *base*)

Definition at line 84 of file utils.c.

```
{
    unsigned long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtoul(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res > UINT_MAX) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.19.2.23 ssize_t getcmdline (char ** *line*, size_t * *len*, FILE * *in*)

Definition at line 733 of file utils.c.

References cmdlineno, and realloc.

```
{
    ssize_t cc;
    char *cp;

    if ((cc = getline(linep, lenp, in)) < 0) {
        return cc;    /* eof or error */
    }
    ++cmdlineno;

    cp = strchr(*linep, '#');
    if (cp) {
        *cp = '\0';
    }

    while ((cp = strstr(*linep, "\\\n")) != NULL) {
        char *line1 = NULL;
        size_t len1 = 0;
        ssize_t cc1;
```

```
        if ((cc1 = getline(&line1, &len1, in)) < 0) {
            fprintf(stderr, "Missing continuation line\n");
            return cc1;
        }

        ++cmdlineno;
        *cp = 0;

        cp = strchr(line1, '#');
        if (cp) {
            *cp = '\0';
        }

        *lenp = strlen(*linep) + strlen(line1) + 1;
        *linep = realloc(*linep, *lenp);
        if (!*linep) {
            fprintf(stderr, "Out of memory\n");
            *lenp = 0;
            return -1;
        }
        cc += cc1 - 2;
        strcat(*linep, line1);
        free(line1);
    }
    return cc;
}
```

7.19.2.24 __u8* hexstring_a2n (const char * *str*, __u8 * *buf*, int *blen*)

Definition at line 674 of file utils.c.

```
{
    int cnt = 0;

    for (;;) {
        unsigned acc;
        char ch;

        acc = 0;

        while ((ch = *str) != ':' && ch != 0) {
            if (ch >= '0' && ch <= '9') {
                ch -= '0';
            } else
                if (ch >= 'a' && ch <= 'f') {
                    ch -= 'a'-10;
                } else
                    if (ch >= 'A' && ch <= 'F') {
                        ch -= 'A'-10;
                    } else {
                        return NULL;
                    }
            acc = (acc<<4) + ch;
            str++;
        }

        if (acc > 255) {
            return NULL;
        }
        if (cnt < blen) {
            buf[cnt] = acc;
            cnt++;
        }
        if (ch == 0) {
            break;
        }
        ++str;
    }
    if (cnt < blen) {
        memset(buf+cnt, 0, blen-cnt);
    }
    return buf;
}
```

7.19.2.25 char* hexstring_n2a (const __u8 * *str*, int *len*, char * *buf*, int *blen*)

Definition at line 655 of file utils.c.

```
                                                                    {
    char *ptr = buf;
    int i;

    for (i=0; i<len; i++) {
        if (blen < 3) {
            break;
        }
        sprintf(ptr, "%02x", str[i]);
        ptr += 2;
        blen -= 2;
        if (i != len-1 && blen > 1) {
            *ptr++ = ':';
            blen--;
        }
    }
    return buf;
}
```

7.19.2.26 void incomplete_command (void)

Definition at line 438 of file utils.c.

```
                              {
    fprintf(stderr, "Command line is not complete. Try option \"help\"\n");
    exit(-1);
}
```

7.19.2.27 int inet_addr_match (const inet_prefix * *a*, const inet_prefix * *b*, int *bits*)

Definition at line 471 of file utils.c.

References inet_prefix::data.

```
                                                                        {
    const __u32 *a1 = a->data;
    const __u32 *a2 = b->data;
    int words = bits >> 0x05;

    bits &= 0x1f;

    if (words)
        if (memcmp(a1, a2, words << 2)) {
            return -1;
        }

    if (bits) {
        __u32 w1, w2;
        __u32 mask;

        w1 = a1[words];
        w2 = a2[words];

        mask = htonl((0xffffffff) << (0x20 - bits));

        if ((w1 ^ w2) & mask) {
            return 1;
        }
    }

    return 0;
}
```

7.19.2.28 void invarg (const char * , const char *)

Definition at line 448 of file utils.c.

```
                                          {
    fprintf(stderr, "Error: argument \"%s\" is wrong: %s\n", arg, msg);
    exit(-1);
}
```

7.19.2.29 int iplink_parse (int *argc,* char ** *argv,* struct iplink_req * *req,* char ** *name,* char ** *type,* char ** *link,* char ** *dev,* int * *group*)

7.19.2.30 const char* ipx_ntop (int *af,* const void * *addr,* char * *str,* size_t *len*)

Definition at line 64 of file ipx_ntop.c.

Referenced by rt_addr_n2a().

```
{
    switch(af) {
        case AF_IPX:
            errno = 0;
            return ipx_ntop1((struct ipx_addr *)addr, str, len);
        default
                :
            errno = EAFNOSUPPORT;
    }

    return NULL;
}
```

7.19.2.31 int ipx_pton (int *af,* const char * *src,* void * *addr*)

Definition at line 101 of file ipx_pton.c.

```
{
    int err;

    switch (af) {
        case AF_IPX:
            errno = 0;
            err = ipx_pton1(src, (struct ipx_addr *)addr);
            break;
        default
                :
            errno = EAFNOSUPPORT;
            err = -1;
    }

    return err;
}
```

7.19.2.32 int makeargs (char * *line,* char * *argv[],* int *maxargs*)

Definition at line 780 of file utils.c.

```
{
    static const char ws[] = " \t\r\n";
    char *cp;
    int argc = 0;

    for (cp = strtok(line, ws); cp; cp = strtok(NULL, ws)) {
        if (argc >= (maxargs - 1)) {
            fprintf(stderr, "Too many arguments to command\n");
            exit(1);
        }
        argv[argc++] = cp;
    }
    argv[argc] = NULL;

    return argc;
}
```

7.19.2.33 int mask2bits (__u32 *netmask*)

Definition at line 48 of file utils.c.

```
                                {
    unsigned bits = 0;
    __u32 mask = ntohl(netmask);
    __u32 host = ~mask;

    /* a valid netmask must be 2^n - 1 */
    if ((host & (host + 1)) != 0) {
        return -1;
    }

    for (; mask; mask <<= 1) {
        ++bits;
    }
    return bits;
}
```

7.19.2.34 int matches (const char * *arg*, const char * *pattern*)

Definition at line 463 of file utils.c.

```
                                                        {
    int len = strlen(cmd);
    if (len > strlen(pattern)) {
        return -1;
    }
    return memcmp(pattern, cmd, len);
}
```

7.19.2.35 void missarg (const char *)

Definition at line 443 of file utils.c.

```
                                {
    fprintf(stderr, "Error: argument \"%s\" is required\n", key);
    exit(-1);
}
```

7.19.2.36 int print_timestamp (FILE * *fp*)

Definition at line 717 of file utils.c.

```
                                {
    struct timeval tv;
    char *tstr;

    memset(&tv, 0, sizeof(tv));
    gettimeofday(&tv, NULL);

    tstr = asctime(localtime(&tv.tv_sec));
    tstr[strlen(tstr)-1] = 0;
    fprintf(fp, "Timestamp: %s %lu usec\n", tstr, tv.tv_usec);
    return 0;
}
```

7.19.2.37 const char* rt_addr_n2a (int *af*, int *len*, const void * *addr*, char * *buf*, int *buflen*)

Definition at line 541 of file utils.c.

References dn_naddr::a_addr, AF_DECnet, dnet_ntop(), and ipx_ntop().

Referenced by format_host().

```
        {
    switch (af) {
        case AF_INET:
```

```
        case AF_INET6:
            return inet_ntop(af, addr, buf, buflen);
        case AF_IPX:
            return ipx_ntop(af, addr, buf, buflen);
        case AF_DECnet: {
            struct dn_naddr dna = { 2, { 0, 0, }};
            memcpy(dna.a_addr, addr, 2);
            return dnet_ntop(af, &dna, buf, buflen);
        }
        default
                :
            return "???";
    }
}
```

7.19.3 Variable Documentation

7.19.3.1 int __iproute2_hz_internal

Definition at line 500 of file utils.c.

7.19.3.2 int __iproute2_user_hz_internal

Definition at line 535 of file utils.c.

7.19.3.3 char* _SL_

7.19.3.4 int cmdlineno

Definition at line 730 of file utils.c.

Referenced by getcmdline().

7.19.3.5 int max_flush_loops

7.19.3.6 int oneline

7.19.3.7 int preferred_family

7.19.3.8 int resolve_hosts

Referenced by format_host().

7.19.3.9 int show_details

7.19.3.10 int show_raw

7.19.3.11 int show_stats

7.19.3.12 int timestamp

7.20 src/libnetlink/inet_proto.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <syslog.h>
#include <fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <string.h>
#include "utils.h"
```

Functions

- char * inet_proto_n2a (int proto, char *buf, int len)
- int inet_proto_a2n (char *buf)

7.20.1 Function Documentation

7.20.1.1 int inet_proto_a2n (char * *buf*)

Definition at line 45 of file inet_proto.c.

References get_u8().

```
                              {
    static char ncache[16];
    static int icache = -1;
    struct protoent *pe;

    if (icache>=0 && strcmp(ncache, buf) == 0) {
        return icache;
    }

    if (buf[0] >= '0' && buf[0] <= '9') {
        __u8 ret;
        if (get_u8(&ret, buf, 10)) {
            return -1;
        }
        return ret;
    }

    pe = getprotobyname(buf);
    if (pe) {
        icache = pe->p_proto;
        strncpy(ncache, pe->p_name, 16);
        return pe->p_proto;
    }
    return -1;
}
```

7.20.1.2 char* inet_proto_n2a (int *proto,* char * *buf,* int *len*)

Definition at line 25 of file inet_proto.c.

```
                                                    {
    static char ncache[16];
    static int icache = -1;
    struct protoent *pe;

    if (proto == icache) {
        return ncache;
    }
```

```
    pe = getprotobynumber(proto);
    if (pe) {
        icache = proto;
        strncpy(ncache, pe->p_name, 16);
        strncpy(buf, pe->p_name, len);
        return buf;
    }
    snprintf(buf, len, "ipproto-%d", proto);
    return buf;
}
```

7.21 src/libnetlink/ipx_ntop.c File Reference

```
#include <errno.h>
#include <sys/types.h>
#include <netinet/in.h>
#include "utils.h"
```

Functions

- const char * ipx_ntop (int af, const void *addr, char *str, size_t len)

7.21.1 Function Documentation

7.21.1.1 const char* ipx_ntop (int *af*, const void * *addr*, char * *str*, size_t *len*)

Definition at line 64 of file ipx_ntop.c.

Referenced by rt_addr_n2a().

```
{
    switch(af) {
        case AF_IPX:
            errno = 0;
            return ipx_ntop1((struct ipx_addr *)addr, str, len);
        default
                :
            errno = EAFNOSUPPORT;
    }

    return NULL;
}
```

7.22 src/libnetlink/ipx_pton.c File Reference

```
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include "utils.h"
```

Functions

- int ipx_pton (int af, const char *src, void *addr)

7.22.1 Function Documentation

7.22.1.1 int ipx_pton (int *af*, const char * *src*, void * *addr*)

Definition at line 101 of file ipx_pton.c.

```
                                                {
    int err;

    switch (af) {
        case AF_IPX:
            errno = 0;
            err = ipx_pton1(src, (struct ipx_addr *)addr);
            break;
        default
                :
            errno = EAFNOSUPPORT;
            err = -1;
    }

    return err;
}
```

7.23 src/libnetlink/libnetlink.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <syslog.h>
#include <fcntl.h>
#include <net/if_arp.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <errno.h>
#include <time.h>
#include <sys/uio.h>
#include "libnetlink.h"
```

Functions

- void rtnl_close (struct rtnl_handle *rth)
- int rtnl_open_byproto (struct rtnl_handle *rth, unsigned subscriptions, int protocol)
- int rtnl_open (struct rtnl_handle *rth, unsigned subscriptions)
- int rtnl_wilddump_request (struct rtnl_handle *rth, int family, int type)
- int rtnl_send (struct rtnl_handle *rth, const void *buf, int len)
- int rtnl_send_check (struct rtnl_handle *rth, const void *buf, int len)
- int rtnl_dump_request (struct rtnl_handle *rth, int type, void *req, int len)
- int rtnl_dump_filter_l (struct rtnl_handle *rth, const struct rtnl_dump_filter_arg *arg)
- int rtnl_dump_filter (struct rtnl_handle *rth, rtnl_filter_t filter, void *arg1)
- int rtnl_talk (struct rtnl_handle *rtnl, struct nlmsghdr *n, pid_t peer, unsigned groups, struct nlmsghdr *answer)
- int rtnl_listen (struct rtnl_handle *rtnl, rtnl_filter_t handler, void *jarg)
- int rtnl_from_file (FILE *rtnl, rtnl_filter_t handler, void *jarg)
- int addattr (struct nlmsghdr *n, int maxlen, int type)
- int addattr8 (struct nlmsghdr *n, int maxlen, int type, __u8 data)
- int addattr16 (struct nlmsghdr *n, int maxlen, int type, __u16 data)
- int addattr32 (struct nlmsghdr *n, int maxlen, int type, __u32 data)
- int addattr64 (struct nlmsghdr *n, int maxlen, int type, __u64 data)

- int addattrstrz (struct nlmsghdr *n, int maxlen, int type, const char *str)
- int addattr_l (struct nlmsghdr *n, int maxlen, int type, const void *data, int alen)
- int addraw_l (struct nlmsghdr *n, int maxlen, const void *data, int len)
- struct rtattr * addattr_nest (struct nlmsghdr *n, int maxlen, int type)
- int addattr_nest_end (struct nlmsghdr *n, struct rtattr *nest)
- struct rtattr * addattr_nest_compat (struct nlmsghdr *n, int maxlen, int type, const void *data, int len)
- int addattr_nest_compat_end (struct nlmsghdr *n, struct rtattr *start)
- int rta_addattr32 (struct rtattr *rta, int maxlen, int type, __u32 data)
- int rta_addattr_l (struct rtattr *rta, int maxlen, int type, const void *data, int alen)
- int parse_rtattr (struct rtattr *tb[], int max, struct rtattr *rta, int len)
- int parse_rtattr_byindex (struct rtattr *tb[], int max, struct rtattr *rta, int len)
- int __parse_rtattr_nested_compat (struct rtattr *tb[], int max, struct rtattr *rta, int len)

Variables

- int rcvbuf = 1024 * 1024

7.23.1 Function Documentation

7.23.1.1 int __parse_rtattr_nested_compat (struct rtattr * *tb[],* int *max,* struct rtattr * *rta,* int *len*)

Definition at line 673 of file libnetlink.c.

References parse_rtattr_nested.

```
                                                {
    if (RTA_PAYLOAD(rta) < len) {
        return -1;
    }
    if (RTA_PAYLOAD(rta) >= RTA_ALIGN(len) + sizeof(struct rtattr)) {
        rta = RTA_DATA(rta) + RTA_ALIGN(len);
        return parse_rtattr_nested(tb, max, rta);
    }
    memset(tb, 0, sizeof(struct rtattr *) * (max + 1));
    return 0;
}
```

7.23.1.2 int addattr (struct nlmsghdr * *n,* int *maxlen,* int *type*)

Definition at line 528 of file libnetlink.c.

References addattr_l().

```
                                                        {
    return addattr_l(n, maxlen, type, NULL, 0);
}
```

7.23.1.3 int addattr16 (struct nlmsghdr * *n,* int *maxlen,* int *type,* __u16 *data*)

Definition at line 536 of file libnetlink.c.

References addattr_l().

```
                                                                {
    return addattr_l(n, maxlen, type, &data, sizeof(__u16));
}
```

7.23.1.4 int addattr32 (struct nlmsghdr * *n*, int *maxlen*, int *type*, __u32 *data*)

Definition at line 540 of file libnetlink.c.

References addattr_l().

Referenced by create_kernmac(), and set_interface_ipaddr().

```
                                                                    {
    return addattr_l(n, maxlen, type, &data, sizeof(__u32));
}
```

7.23.1.5 int addattr64 (struct nlmsghdr * *n*, int *maxlen*, int *type*, __u64 *data*)

Definition at line 544 of file libnetlink.c.

References addattr_l().

```
                                                                    {
    return addattr_l(n, maxlen, type, &data, sizeof(__u64));
}
```

7.23.1.6 int addattr8 (struct nlmsghdr * *n*, int *maxlen*, int *type*, __u8 *data*)

Definition at line 532 of file libnetlink.c.

References addattr_l().

```
                                                                  {
    return addattr_l(n, maxlen, type, &data, sizeof(__u8));
}
```

7.23.1.7 int addattr_l (struct nlmsghdr * *n*, int *maxlen*, int *type*, const void * *data*, int *alen*)

Definition at line 552 of file libnetlink.c.

References NLMSG_TAIL.

Referenced by addattr(), addattr16(), addattr32(), addattr64(), addattr8(), addattr_nest(), addattr_nest_compat(), addattrstrz(), create_kernmac(), create_kernvlan(), set_interface_addr(), set_interface_ipaddr(), and set_interface_name().

```
                      {
    int len = RTA_LENGTH(alen);
    struct rtattr *rta;

    if (NLMSG_ALIGN(n->nlmsg_len) + RTA_ALIGN(len) > maxlen) {
        fprintf(stderr, "addattr_l ERROR: message exceeded bound of %d\n",
      maxlen);
        return -1;
    }
    rta = NLMSG_TAIL(n);
    rta->rta_type = type;
    rta->rta_len = len;
    memcpy(RTA_DATA(rta), data, alen);
    n->nlmsg_len = NLMSG_ALIGN(n->nlmsg_len) + RTA_ALIGN(len);
    return 0;
}
```

7.23.1.8 struct rtattr* addattr_nest (struct nlmsghdr * *n*, int *maxlen*, int *type*) [read]

Definition at line 581 of file libnetlink.c.

References addattr_l(), and NLMSG_TAIL.

Referenced by addattr_nest_compat().

```
                                                                    {
    struct rtattr *nest = NLMSG_TAIL(n);

    addattr_l(n, maxlen, type, NULL, 0);
    return nest;
}
```

7.23.1.9 struct rtattr* addattr_nest_compat (struct nlmsghdr * *n*, int *maxlen*, int *type*, const void * *data*, int *len*) [read]

Definition at line 593 of file libnetlink.c.

References addattr_l(), addattr_nest(), and NLMSG_TAIL.

```
                                                                {
    struct rtattr *start = NLMSG_TAIL(n);

    addattr_l(n, maxlen, type, data, len);
    addattr_nest(n, maxlen, type);
    return start;
}
```

7.23.1.10 int addattr_nest_compat_end (struct nlmsghdr * *n*, struct rtattr * *start*)

Definition at line 602 of file libnetlink.c.

References addattr_nest_end(), and NLMSG_TAIL.

```
                                                                    {
    struct rtattr *nest = (void *)start + NLMSG_ALIGN(start->rta_len);

    start->rta_len = (void *)NLMSG_TAIL(n) - (void *)start;
    addattr_nest_end(n, nest);
    return n->nlmsg_len;
}
```

7.23.1.11 int addattr_nest_end (struct nlmsghdr * *n*, struct rtattr * *nest*)

Definition at line 588 of file libnetlink.c.

References NLMSG_TAIL.

Referenced by addattr_nest_compat_end().

```
                                                                {
    nest->rta_len = (void *)NLMSG_TAIL(n) - (void *)nest;
    return n->nlmsg_len;
}
```

7.23.1.12 int addattrstrz (struct nlmsghdr * *n*, int *maxlen*, int *type*, const char * *str*)

Definition at line 548 of file libnetlink.c.

References addattr_l().

```
                                                                        {
    return addattr_l(n, maxlen, type, str, strlen(str)+1);
}
```

7.23.1.13 int addraw_l (struct nlmsghdr * *n*, int *maxlen*, const void * *data*, int *len*)

Definition at line 569 of file libnetlink.c.

References NLMSG_TAIL.

```
                                                                        {
    if (NLMSG_ALIGN(n->nlmsg_len) + NLMSG_ALIGN(len) > maxlen) {
        fprintf(stderr, "addraw_l ERROR: message exceeded bound of %d\n",maxlen
      );
        return -1;
    }

    memcpy(NLMSG_TAIL(n), data, len);
    memset((void *) NLMSG_TAIL(n) + len, 0, NLMSG_ALIGN(len) - len);
    n->nlmsg_len = NLMSG_ALIGN(n->nlmsg_len) + NLMSG_ALIGN(len);
    return 0;
}
```

7.23.1.14 int parse_rtattr (struct rtattr * *tb[]*, int *max*, struct rtattr * *rta*, int *len*)

Definition at line 643 of file libnetlink.c.

Referenced by ll_remember_index().

```
                                                                                {
    memset(tb, 0, sizeof(struct rtattr *) * (max + 1));
    while (RTA_OK(rta, len)) {
        if ((rta->rta_type <= max) && (!tb[rta->rta_type])) {
            tb[rta->rta_type] = rta;
        }
        rta = RTA_NEXT(rta,len);
    }
    if (len) {
        fprintf(stderr, "!!!Deficit %d, rta_len=%d\n", len, rta->rta_len);
    }
    return 0;
}
```

7.23.1.15 int parse_rtattr_byindex (struct rtattr * *tb[]*, int *max*, struct rtattr * *rta*, int *len*)

Definition at line 657 of file libnetlink.c.

```
           {
    int i = 0;

    memset(tb, 0, sizeof(struct rtattr *) * max);
    while (RTA_OK(rta, len)) {
        if (rta->rta_type <= max && i < max) {
            tb[i++] = rta;
        }
        rta = RTA_NEXT(rta,len);
    }
    if (len) {
        fprintf(stderr, "!!!Deficit %d, rta_len=%d\n", len, rta->rta_len);
    }
    return i;
}
```

7.23.1.16 int rta_addattr32 (struct rtattr * *rta*, int *maxlen*, int *type*, __u32 *data*)

Definition at line 610 of file libnetlink.c.

```
                                                                        {
    int len = RTA_LENGTH(4);
    struct rtattr *subrta;

    if (RTA_ALIGN(rta->rta_len) + len > maxlen) {
```

```
        fprintf(stderr,"rta_addattr32: Error! max allowed bound %d exceeded\n",
      maxlen);
        return -1;
    }
    subrta = (struct rtattr *)(((char *)rta) + RTA_ALIGN(rta->rta_len));
    subrta->rta_type = type;
    subrta->rta_len = len;
    memcpy(RTA_DATA(subrta), &data, 4);
    rta->rta_len = NLMSG_ALIGN(rta->rta_len) + len;
    return 0;
}
```

7.23.1.17 int rta_addattr_l (struct rtattr * *rta*, int *maxlen*, int *type*, const void * *data*, int *alen*)

Definition at line 626 of file libnetlink.c.

```
                                                {
    struct rtattr *subrta;
    int len = RTA_LENGTH(alen);

    if (RTA_ALIGN(rta->rta_len) + RTA_ALIGN(len) > maxlen) {
        fprintf(stderr,"rta_addattr_l: Error! max allowed bound %d exceeded\n",
      maxlen);
        return -1;
    }
    subrta = (struct rtattr *)(((char *)rta) + RTA_ALIGN(rta->rta_len));
    subrta->rta_type = type;
    subrta->rta_len = len;
    memcpy(RTA_DATA(subrta), data, alen);
    rta->rta_len = NLMSG_ALIGN(rta->rta_len) + RTA_ALIGN(len);
    return 0;
}
```

7.23.1.18 void rtnl_close (struct rtnl_handle * *rth*)

Definition at line 30 of file libnetlink.c.

References rtnl_handle::fd.

```
                                            {
    if (rth->fd >= 0) {
        close(rth->fd);
        rth->fd = -1;
    }
}
```

7.23.1.19 int rtnl_dump_filter (struct rtnl_handle * *rth*, rtnl_filter_t *filter*, void * *arg1*)

Definition at line 264 of file libnetlink.c.

References rtnl_dump_filter_arg::arg1, rtnl_dump_filter_arg::filter, and rtnl_dump_filter_l().

Referenced by ll_init_map().

```
                                    {
    const struct rtnl_dump_filter_arg a[2] = {
        { .filter = filter, .arg1 = arg1, },
        { .filter = NULL,   .arg1 = NULL, },
    };

    return rtnl_dump_filter_l(rth, a);
}
```

7.23.1.20 int rtnl_dump_filter_l (struct rtnl_handle * *rth*, const struct rtnl_dump_filter_arg * *arg*)

Definition at line 175 of file libnetlink.c.

References rtnl_dump_filter_arg::arg1, rtnl_handle::dump, rtnl_handle::fd, rtnl_dump_filter_arg::filter, and rtnl_handle::local.

Referenced by rtnl_dump_filter().

```
                                                                {
    struct sockaddr_nl nladdr;
    struct iovec iov;
    struct msghdr msg = {
        .msg_name = &nladdr,
        .msg_namelen = sizeof(nladdr),
        .msg_iov = &iov,
        .msg_iovlen = 1,
    };
    char buf[16384];

    iov.iov_base = buf;
    while (1) {
        int status;
        const struct rtnl_dump_filter_arg *a;
        int found_done = 0;
        int msglen = 0;

        iov.iov_len = sizeof(buf);
        status = recvmsg(rth->fd, &msg, 0);

        if (status < 0) {
            if (errno == EINTR || errno == EAGAIN) {
                continue;
            }
            fprintf(stderr, "netlink receive error %s (%d)\n",
                    strerror(errno), errno);
            return -1;
        }

        if (status == 0) {
            fprintf(stderr, "EOF on netlink\n");
            return -1;
        }

        for (a = arg; a->filter; a++) {
            struct nlmsghdr *h = (struct nlmsghdr *)buf;
            msglen = status;

            while (NLMSG_OK(h, msglen)) {
                int err;

                if (nladdr.nl_pid != 0 ||
                        h->nlmsg_pid != rth->local.nl_pid ||
                        h->nlmsg_seq != rth->dump) {
                    goto skip_it;
                }

                if (h->nlmsg_type == NLMSG_DONE) {
                    found_done = 1;
                    break; /* process next filter */
                }
                if (h->nlmsg_type == NLMSG_ERROR) {
                    struct nlmsgerr *err = (struct nlmsgerr *)NLMSG_DATA(h);
                    if (h->nlmsg_len < NLMSG_LENGTH(sizeof(struct nlmsgerr))) {
                        fprintf(stderr,
                                "ERROR truncated\n");
                    } else {
                        errno = -err->error;
                        perror("RTNETLINK answers");
                    }
                    return -1;
                }
                err = a->filter(&nladdr, h, a->arg1);
                if (err < 0) {
                    return err;
                }

skip_it:
                h = NLMSG_NEXT(h, msglen);
            }
        }

        if (found_done) {
            return 0;
        }

        if (msg.msg_flags & MSG_TRUNC) {
            fprintf(stderr, "Message truncated\n");
            continue;
        }
```

```
        if (msglen) {
            fprintf(stderr, "!!!Remnant of size %d\n", msglen);
            exit(1);
        }
    }
}
```

7.23.1.21 int rtnl_dump_request (struct rtnl_handle * *rth,* int *type,* void * *req,* int *len*)

Definition at line 152 of file libnetlink.c.

References rtnl_handle::dump, rtnl_handle::fd, and rtnl_handle::seq.

```
{
    struct nlmsghdr nlh;
    struct sockaddr_nl nladdr = { .nl_family = AF_NETLINK };
    struct iovec iov[2] = {
        { .iov_base = &nlh, .iov_len = sizeof(nlh) },
        { .iov_base = req, .iov_len = len }
    };
    struct msghdr msg = {
        .msg_name = &nladdr,
        .msg_namelen =  sizeof(nladdr),
        .msg_iov = iov,
        .msg_iovlen = 2,
    };

    nlh.nlmsg_len = NLMSG_LENGTH(len);
    nlh.nlmsg_type = type;
    nlh.nlmsg_flags = NLM_F_DUMP|NLM_F_REQUEST;
    nlh.nlmsg_pid = 0;
    nlh.nlmsg_seq = rth->dump = ++rth->seq;

    return sendmsg(rth->fd, &msg, 0);
}
```

7.23.1.22 int rtnl_from_file (FILE * *rtnl,* rtnl_filter_t *handler,* void * *jarg*)

Definition at line 472 of file libnetlink.c.

```
{
    int status;
    struct sockaddr_nl nladdr;
    char   buf[8192];
    struct nlmsghdr *h = (void *)buf;

    memset(&nladdr, 0, sizeof(nladdr));
    nladdr.nl_family = AF_NETLINK;
    nladdr.nl_pid = 0;
    nladdr.nl_groups = 0;

    while (1) {
        int err, len;
        int l;

        status = fread(&buf, 1, sizeof(*h), rtnl);

        if (status < 0) {
            if (errno == EINTR) {
                continue;
            }
            perror("rtnl_from_file: fread");
            return -1;
        }
        if (status == 0) {
            return 0;
        }

        len = h->nlmsg_len;
        l = len - sizeof(*h);

        if (l<0 || len>sizeof(buf)) {
            fprintf(stderr, "!!!malformed message: len=%d @%lu\n",
                    len, ftell(rtnl));
            return -1;
        }
```

```
        status = fread(NLMSG_DATA(h), 1, NLMSG_ALIGN(l), rtnl);

        if (status < 0) {
            perror("rtnl_from_file: fread");
            return -1;
        }
        if (status < l) {
            fprintf(stderr, "rtnl-from_file: truncated message\n");
            return -1;
        }

        err = handler(&nladdr, h, jarg);
        if (err < 0) {
            return err;
        }
    }
}
```

7.23.1.23 int rtnl_listen (struct rtnl_handle * *rtnl*, rtnl_filter_t *handler*, void * *jarg*)

Definition at line 395 of file libnetlink.c.

References rtnl_handle::fd.

```
                        {
    int status;
    struct nlmsghdr *h;
    struct sockaddr_nl nladdr;
    struct iovec iov;
    struct msghdr msg = {
        .msg_name = &nladdr,
        .msg_namelen = sizeof(nladdr),
        .msg_iov = &iov,
        .msg_iovlen = 1,
    };
    char   buf[8192];

    memset(&nladdr, 0, sizeof(nladdr));
    nladdr.nl_family = AF_NETLINK;
    nladdr.nl_pid = 0;
    nladdr.nl_groups = 0;

    iov.iov_base = buf;
    while (1) {
        iov.iov_len = sizeof(buf);
        status = recvmsg(rtnl->fd, &msg, 0);

        if (status < 0) {
            if (errno == EINTR || errno == EAGAIN) {
                continue;
            }
            fprintf(stderr, "netlink receive error %s (%d)\n",
                    strerror(errno), errno);
            if (errno == ENOBUFS) {
                continue;
            }
            return -1;
        }
        if (status == 0) {
            fprintf(stderr, "EOF on netlink\n");
            return -1;
        }
        if (msg.msg_namelen != sizeof(nladdr)) {
            fprintf(stderr, "Sender address length == %d\n", msg.msg_namelen);
            exit(1);
        }
        for (h = (struct nlmsghdr *)buf; status >= sizeof(*h); ) {
            int err;
            int len = h->nlmsg_len;
            int l = len - sizeof(*h);

            if (l<0 || len>status) {
                if (msg.msg_flags & MSG_TRUNC) {
                    fprintf(stderr, "Truncated message\n");
                    return -1;
                }
                fprintf(stderr, "!!!malformed message: len=%d\n", len);
                exit(1);
            }

            err = handler(&nladdr, h, jarg);
            if (err < 0) {
```

```
                return err;
            }

            status -= NLMSG_ALIGN(len);
            h = (struct nlmsghdr *)((char *)h + NLMSG_ALIGN(len));
        }
        if (msg.msg_flags & MSG_TRUNC) {
            fprintf(stderr, "Message truncated\n");
            continue;
        }
        if (status) {
            fprintf(stderr, "!!!Remnant of size %d\n", status);
            exit(1);
        }
    }
}
```

7.23.1.24 int rtnl_open (struct rtnl_handle * *rth*, unsigned *subscriptions*)

Definition at line 85 of file libnetlink.c.

References rtnl_open_byproto().

```
                                                                    {
    return rtnl_open_byproto(rth, subscriptions, NETLINK_ROUTE
      );
}
```

7.23.1.25 int rtnl_open_byproto (struct rtnl_handle * *rth*, unsigned *subscriptions*, int *protocol*)

Definition at line 37 of file libnetlink.c.

References rtnl_handle::fd, rtnl_handle::local, rcvbuf, and rtnl_handle::seq.

Referenced by rtnl_open().

```
                                    {
    socklen_t addr_len;
    int sndbuf = 32768;

    memset(rth, 0, sizeof(*rth));

    rth->fd = socket(AF_NETLINK, SOCK_RAW, protocol);
    if (rth->fd < 0) {
        perror("Cannot open netlink socket");
        return -1;
    }

    if (setsockopt(rth->fd,SOL_SOCKET,SO_SNDBUF,&sndbuf,sizeof(sndbuf)) < 0)
      {
        perror("SO_SNDBUF");
        return -1;
    }

    if (setsockopt(rth->fd,SOL_SOCKET,SO_RCVBUF,&rcvbuf,sizeof(rcvbuf
      )) < 0) {
        perror("SO_RCVBUF");
        return -1;
    }

    memset(&rth->local, 0, sizeof(rth->local));
    rth->local.nl_family = AF_NETLINK;
    rth->local.nl_groups = subscriptions;

    if (bind(rth->fd, (struct sockaddr *)&rth->local, sizeof(rth->local
      )) < 0) {
        perror("Cannot bind netlink socket");
        return -1;
    }
    addr_len = sizeof(rth->local);
    if (getsockname(rth->fd, (struct sockaddr *)&rth->local, &addr_len)
      < 0) {
        perror("Cannot getsockname");
        return -1;
    }
    if (addr_len != sizeof(rth->local)) {
```

```
        fprintf(stderr, "Wrong address length %d\n", addr_len);
        return -1;
    }
    if (rth->local.nl_family != AF_NETLINK) {
        fprintf(stderr, "Wrong address family %d\n", rth->local.nl_family)
      ;
        return -1;
    }
    rth->seq = time(NULL);
    return 0;
}
```

7.23.1.26 int rtnl_send (struct rtnl_handle * *rth,* const void * *buf,* int *len*)

Definition at line 113 of file libnetlink.c.

References rtnl_handle::fd.

```
                                                                    {
    return send(rth->fd, buf, len, 0);
}
```

7.23.1.27 int rtnl_send_check (struct rtnl_handle * *rth,* const void * *buf,* int *len*)

Definition at line 117 of file libnetlink.c.

References rtnl_handle::fd.

```
                                                                        {
    struct nlmsghdr *h;
    int status;
    char resp[1024];

    status = send(rth->fd, buf, len, 0);
    if (status < 0) {
        return status;
    }

    /* Check for immediate errors */
    status = recv(rth->fd, resp, sizeof(resp), MSG_DONTWAIT|MSG_PEEK);
    if (status < 0) {
        if (errno == EAGAIN) {
            return 0;
        }
        return -1;
    }

    for (h = (struct nlmsghdr *)resp; NLMSG_OK(h, status);
            h = NLMSG_NEXT(h, status)) {
        if (h->nlmsg_type == NLMSG_ERROR) {
            struct nlmsgerr *err = (struct nlmsgerr *)NLMSG_DATA(h);
            if (h->nlmsg_len < NLMSG_LENGTH(sizeof(struct nlmsgerr))) {
                fprintf(stderr, "ERROR truncated\n");
            } else {
                errno = -err->error;
            }
            return -1;
        }
    }

    return 0;
}
```

7.23.1.28 int rtnl_talk (struct rtnl_handle * *rtnl,* struct nlmsghdr * *n,* pid_t *peer,* unsigned *groups,* struct nlmsghdr * *answer*)

Definition at line 275 of file libnetlink.c.

References rtnl_handle::fd, rtnl_handle::local, and rtnl_handle::seq.

Referenced by create_kernmac(), create_kernvlan(), set_interface_addr(), set_interface_flags(), set_interface_ipaddr(), and set_interface_name().

```
                                                {
    int status;
    unsigned seq;
    struct nlmsghdr *h;
    struct sockaddr_nl nladdr;
    struct iovec iov = {
        .iov_base = (void *) n,
        .iov_len = n->nlmsg_len
    };
    struct msghdr msg = {
        .msg_name = &nladdr,
        .msg_namelen = sizeof(nladdr),
        .msg_iov = &iov,
        .msg_iovlen = 1,
    };
    char   buf[16384];

    memset(&nladdr, 0, sizeof(nladdr));
    nladdr.nl_family = AF_NETLINK;
    nladdr.nl_pid = peer;
    nladdr.nl_groups = groups;

    n->nlmsg_seq = seq = ++rtnl->seq;

    if (answer == NULL) {
        n->nlmsg_flags |= NLM_F_ACK;
    }

    status = sendmsg(rtnl->fd, &msg, 0);

    if (status < 0) {
        perror("Cannot talk to rtnetlink");
        return -1;
    }

    memset(buf,0,sizeof(buf));

    iov.iov_base = buf;

    while (1) {
        iov.iov_len = sizeof(buf);
        status = recvmsg(rtnl->fd, &msg, 0);

        if (status < 0) {
            if (errno == EINTR || errno == EAGAIN) {
                continue;
            }
            fprintf(stderr, "netlink receive error %s (%d)\n",
                    strerror(errno), errno);
            return -1;
        }
        if (status == 0) {
            fprintf(stderr, "EOF on netlink\n");
            return -1;
        }
        if (msg.msg_namelen != sizeof(nladdr)) {
            fprintf(stderr, "sender address length == %d\n", msg.msg_namelen);
            exit(1);
        }
        for (h = (struct nlmsghdr *)buf; status >= sizeof(*h); ) {
            int len = h->nlmsg_len;
            int l = len - sizeof(*h);

            if (l < 0 || len>status) {
                if (msg.msg_flags & MSG_TRUNC) {
                    fprintf(stderr, "Truncated message\n");
                    return -1;
                }
                fprintf(stderr, "!!!malformed message: len=%d\n", len);
                exit(1);
            }

            if (nladdr.nl_pid != peer ||
                    h->nlmsg_pid != rtnl->local.nl_pid ||
                    h->nlmsg_seq != seq) {
                /* Don't forget to skip that message. */
                status -= NLMSG_ALIGN(len);
                h = (struct nlmsghdr *)((char *)h + NLMSG_ALIGN(len));
                continue;
            }

            if (h->nlmsg_type == NLMSG_ERROR) {
                struct nlmsgerr *err = (struct nlmsgerr *)NLMSG_DATA(h);
                if (l < sizeof(struct nlmsgerr)) {
                    fprintf(stderr, "ERROR truncated\n");
                } else {
                    if (!err->error) {
```

```
                    if (answer) {
                        memcpy(answer, h, h->nlmsg_len);
                    }
                    return 0;
                }

                fprintf(stderr, "RTNETLINK answers: %s\n", strerror(-err->
    error));
                errno = -err->error;
            }
            return -1;
        }
        if (answer) {
            memcpy(answer, h, h->nlmsg_len);
            return 0;
        }

        fprintf(stderr, "Unexpected reply!!!\n");

        status -= NLMSG_ALIGN(len);
        h = (struct nlmsghdr *)((char *)h + NLMSG_ALIGN(len));
    }
    if (msg.msg_flags & MSG_TRUNC) {
        fprintf(stderr, "Message truncated\n");
        continue;
    }
    if (status) {
        fprintf(stderr, "!!!Remnant of size %d\n", status);
        exit(1);
    }
  }
}
```

7.23.1.29 int rtnl_wilddump_request (struct rtnl_handle * *rth*, int *family*, int *type*)

Definition at line 89 of file libnetlink.c.

References rtnl_handle::dump, rtnl_handle::fd, and rtnl_handle::seq.

Referenced by ll_init_map().

```
                                                                    {
    struct {
        struct nlmsghdr nlh;
        struct rtgenmsg g;
        __u16 align_rta;    /* attribute has to be 32bit aligned */
        struct rtattr ext_req;
        __u32 ext_filter_mask;
    } req;

    memset(&req, 0, sizeof(req));
    req.nlh.nlmsg_len = sizeof(req);
    req.nlh.nlmsg_type = type;
    req.nlh.nlmsg_flags = NLM_F_DUMP|NLM_F_REQUEST;
    req.nlh.nlmsg_pid = 0;
    req.nlh.nlmsg_seq = rth->dump = ++rth->seq;
    req.g.rtgen_family = family;

    req.ext_req.rta_type = IFLA_EXT_MASK;
    req.ext_req.rta_len = RTA_LENGTH(sizeof(__u32));
    req.ext_filter_mask = RTEXT_FILTER_VF;

    return send(rth->fd, (void *)&req, sizeof(req), 0);
}
```

7.23.2 Variable Documentation

7.23.2.1 int rcvbuf = 1024 * 1024

Definition at line 28 of file libnetlink.c.

Referenced by rtnl_open_byproto().

7.24 src/libnetlink/ll_addr.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <syslog.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <linux/netdevice.h>
#include <linux/if_arp.h>
#include <linux/sockios.h>
#include "rt_names.h"
#include "utils.h"
```

Functions

- const char * ll_addr_n2a (unsigned char *addr, int alen, int type, char *buf, int blen)
- int ll_addr_a2n (char *lladdr, int len, char *arg)

7.24.1 Function Documentation

7.24.1.1 int ll_addr_a2n (char * *lladdr*, int *len*, char * *arg*)

Definition at line 59 of file ll_addr.c.

References inet_prefix::data, and get_addr_1().

```
                                                {
    if (strchr(arg, '.')) {
        inet_prefix pfx;
        if (get_addr_1(&pfx, arg, AF_INET)) {
            fprintf(stderr, "\"%s\" is invalid lladdr.\n", arg);
            return -1;
        }
        if (len < 4) {
            return -1;
        }
        memcpy(lladdr, pfx.data, 4);
        return 4;
    } else {
        int i;

        for (i=0; i<len; i++) {
            int temp;
            char *cp = strchr(arg, ':');
            if (cp) {
                *cp = 0;
                cp++;
            }
            if (sscanf(arg, "%x", &temp) != 1) {
                fprintf(stderr, "\"%s\" is invalid lladdr.\n", arg);
                return -1;
            }
            if (temp < 0 || temp > 255) {
                fprintf(stderr, "\"%s\" is invalid lladdr.\n", arg);
                return -1;
            }
            lladdr[i] = temp;
            if (!cp) {
                break;
            }
            arg = cp;
        }
        return i+1;
```

```
    }
}
```

7.24.1.2 const char* ll_addr_n2a (unsigned char * *addr*, int *alen*, int *type*, char * *buf*, int *blen*)

Definition at line 32 of file ll_addr.c.

```
          {
    int i;
    int l;

    if (alen == 4 &&
            (type == ARPHRD_TUNNEL || type == ARPHRD_SIT || type ==
      ARPHRD_IPGRE)) {
        return inet_ntop(AF_INET, addr, buf, blen);
    }
    if (alen == 16 && type == ARPHRD_TUNNEL6) {
        return inet_ntop(AF_INET6, addr, buf, blen);
    }
    l = 0;
    for (i=0; i<alen; i++) {
        if (i==0) {
            snprintf(buf+l, blen, "%02x", addr[i]);
            blen -= 2;
            l += 2;
        } else {
            snprintf(buf+l, blen, ":%02x", addr[i]);
            blen -= 3;
            l += 3;
        }
    }
    return buf;
}
```

7.25 src/libnetlink/ll_map.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <syslog.h>
#include <fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <linux/if.h>
#include "libnetlink.h"
#include "ll_map.h"
```

Data Structures

- struct ll_cache

Macros

- #define IDXMAP_SIZE 1024

Functions

- unsigned int if_nametoindex (const char *)
- int ll_remember_index (const struct sockaddr_nl *who, struct nlmsghdr *n, void *arg)

- const char * ll_idx_n2a (unsigned idx, char *buf)
- const char * ll_index_to_name (unsigned idx)
- int ll_index_to_type (unsigned idx)
- unsigned ll_index_to_flags (unsigned idx)
- unsigned ll_index_to_addr (unsigned idx, unsigned char *addr, unsigned alen)
- unsigned ll_name_to_index (const char *name)
- int ll_init_map (struct rtnl_handle *rth, int reinit)

7.25.1 Macro Definition Documentation

7.25.1.1 #define IDXMAP_SIZE 1024

Definition at line 38 of file ll_map.c.

Referenced by ll_name_to_index(), and ll_remember_index().

7.25.2 Function Documentation

7.25.2.1 unsigned int if_nametoindex (const char *)

Referenced by ll_name_to_index().

7.25.2.2 const char* ll_idx_n2a (unsigned *idx*, char * *buf*)

Definition at line 99 of file ll_map.c.

References ll_cache::idx_next, ll_cache::index, and ll_cache::name.

Referenced by ll_index_to_name().

```
                                                {
    const struct ll_cache *im;

    if (idx == 0) {
        return "*";
    }

    for (im = idxhead(idx); im; im = im->idx_next)
        if (im->index == idx) {
            return im->name;
        }

    snprintf(buf, IFNAMSIZ, "if%d", idx);
    return buf;
}
```

7.25.2.3 unsigned ll_index_to_addr (unsigned *idx*, unsigned char * *addr*, unsigned *alen*)

Definition at line 149 of file ll_map.c.

References ll_cache::addr, ll_cache::alen, ll_cache::idx_next, and ll_cache::index.

Referenced by ifhwaddr().

```
                                                {
    const struct ll_cache *im;

    if (idx == 0) {
        return 0;
    }

    for (im = idxhead(idx); im; im = im->idx_next) {
        if (im->index == idx) {
            if (alen > sizeof(im->addr)) {
```

```
                alen = sizeof(im->addr);
            }
            if (alen > im->alen) {
                alen = im->alen;
            }
            memcpy(addr, im->addr, alen);
            return alen;
        }
    }
    return 0;
}
```

7.25.2.4 unsigned ll_index_to_flags (unsigned *idx*)

Definition at line 135 of file ll_map.c.

References ll_cache::flags, ll_cache::idx_next, and ll_cache::index.

Referenced by set_interface_flags().

```
                                        {
    const struct ll_cache *im;

    if (idx == 0) {
        return 0;
    }

    for (im = idxhead(idx); im; im = im->idx_next)
        if (im->index == idx) {
            return im->flags;
        }
    return 0;
}
```

7.25.2.5 const char* ll_index_to_name (unsigned *idx*)

Definition at line 116 of file ll_map.c.

References ll_idx_n2a().

```
                                          {
    static char nbuf[IFNAMSIZ];

    return ll_idx_n2a(idx, nbuf);
}
```

7.25.2.6 int ll_index_to_type (unsigned *idx*)

Definition at line 122 of file ll_map.c.

References ll_cache::idx_next, ll_cache::index, and ll_cache::type.

```
                                      {
    const struct ll_cache *im;

    if (idx == 0) {
        return -1;
    }
    for (im = idxhead(idx); im; im = im->idx_next)
        if (im->index == idx) {
            return im->type;
        }
    return -1;
}
```

7.25.2.7 int ll_init_map (struct rtnl_handle * *rth*, int *reinit*)

Definition at line 204 of file ll_map.c.

References ll_remember_index(), rtnl_dump_filter(), and rtnl_wilddump_request().

Referenced by get_iface_index().

```
                                                    {
    static int initialized;

    if (initialized && !reinit) {
        return 0;
    }

    if (rtnl_wilddump_request(rth, AF_UNSPEC, RTM_GETLINK)
       < 0) {
        perror("Cannot send dump request");
        exit(1);
    }

    if (rtnl_dump_filter(rth, ll_remember_index
      , NULL) < 0) {
        fprintf(stderr, "Dump terminated\n");
        exit(1);
    }

    initialized = 1;

    return 0;
}
```

7.25.2.8 unsigned ll_name_to_index (const char * *name*)

Definition at line 172 of file ll_map.c.

References ll_cache::idx_next, IDXMAP_SIZE, if_nametoindex(), ll_cache::index, and ll_cache::name.

Referenced by get_iface_index().

```
                                            {
    static char ncache[IFNAMSIZ];
    static int icache;
    struct ll_cache *im;
    int i;
    unsigned idx;

    if (name == NULL) {
        return 0;
    }

    if (icache && strcmp(name, ncache) == 0) {
        return icache;
    }

    for (i=0; i<IDXMAP_SIZE; i++) {
        for (im = idx_head[i]; im; im = im->idx_next) {
            if (strcmp(im->name, name) == 0) {
                icache = im->index;
                strcpy(ncache, name);
                return im->index;
            }
        }
    }

    idx = if_nametoindex(name);
    if (idx == 0) {
        sscanf(name, "if%u", &idx);
    }
    return idx;
}
```

7.25.2.9 int ll_remember_index (const struct sockaddr_nl * *who*, struct nlmsghdr * *n*, void * *arg*)

Definition at line 45 of file ll_map.c.

References ll_cache::addr, ll_cache::alen, ll_cache::flags, ll_cache::idx_next, IDXMAP_SIZE, IFLA_PAYLOAD, IFLA_RTA, ll_cache::index, malloc, ll_cache::name, parse_rtattr(), and ll_cache::type.

Referenced by ll_init_map().

```
                                                    {
    int h;
    struct ifinfomsg *ifi = NLMSG_DATA(n);
    struct ll_cache *im, **imp;
    struct rtattr *tb[IFLA_MAX+1];

    if (n->nlmsg_type != RTM_NEWLINK) {
        return 0;
    }

    if (n->nlmsg_len < NLMSG_LENGTH(sizeof(ifi))) {
        return -1;
    }

    memset(tb, 0, sizeof(tb));
    parse_rtattr(tb, IFLA_MAX, IFLA_RTA(ifi), IFLA_PAYLOAD
      (n));
    if (tb[IFLA_IFNAME] == NULL) {
        return 0;
    }

    h = ifi->ifi_index & (IDXMAP_SIZE - 1);
    for (imp = &idx_head[h]; (im=*imp)!=NULL; imp = &im->idx_next)
        if (im->index == ifi->ifi_index) {
            break;
        }

    if (im == NULL) {
        im = malloc(sizeof(*im));
        if (im == NULL) {
            return 0;
        }
        im->idx_next = *imp;
        im->index = ifi->ifi_index;
        *imp = im;
    }

    im->type = ifi->ifi_type;
    im->flags = ifi->ifi_flags;
    if (tb[IFLA_ADDRESS]) {
        int alen;
        im->alen = alen = RTA_PAYLOAD(tb[IFLA_ADDRESS]);
        if (alen > sizeof(im->addr)) {
            alen = sizeof(im->addr);
        }
        memcpy(im->addr, RTA_DATA(tb[IFLA_ADDRESS]), alen);
    } else {
        im->alen = 0;
        memset(im->addr, 0, sizeof(im->addr));
    }
    strcpy(im->name, RTA_DATA(tb[IFLA_IFNAME]));
    return 0;
}
```

7.26 src/libnetlink/ll_proto.c File Reference

```
#include <stdio.h>
```

```
#include <stdlib.h>
#include <unistd.h>
#include <syslog.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <linux/netdevice.h>
#include <linux/if_arp.h>
#include <linux/sockios.h>
#include "utils.h"
#include "rt_names.h"
```

Macros

- #define __PF(f, n) { ETH_P_##f, #n },

Functions

- const char * ll_proto_n2a (unsigned short id, char *buf, int len)
- int ll_proto_a2n (unsigned short *id, char *buf)

7.26.1 Macro Definition Documentation

7.26.1.1 #define __PF(*f, n*) { ETH_P_##f, #n },

Definition at line 32 of file ll_proto.c.

Referenced by ll_type_n2a().

7.26.2 Function Documentation

7.26.2.1 int ll_proto_a2n (unsigned short * *id,* char * *buf*)

Definition at line 103 of file ll_proto.c.

References get_u16(), and name.

```
                                                {
    int i;
    for (i=0; i<sizeof(llproto_names)/sizeof(llproto_names[0]); i++) {
        if (strcasecmp(llproto_names[i].name, buf) == 0) {
            *id = htons(llproto_names[i].id);
            return 0;
        }
    }
    if (get_u16(id, buf, 0)) {
        return -1;
    }
    *id = htons(*id);
    return 0;
}
```

7.26.2.2 const char* ll_proto_n2a (unsigned short *id,* char * *buf,* int *len*)

Definition at line 89 of file ll_proto.c.

```
                                                                {
    int i;

    id = ntohs(id);

    for (i=0; i<sizeof(llproto_names)/sizeof(llproto_names[0]); i++) {
        if (llproto_names[i].id == id) {
            return llproto_names[i].name;
        }
    }
    snprintf(buf, len, "[%d]", id);
    return buf;
}
```

7.26.3 Variable Documentation

7.26.3.1 int id

Definition at line 34 of file ll_proto.c.

Referenced by rtnl_dsfield_n2a(), rtnl_rtprot_n2a(), rtnl_rtrealm_n2a(), rtnl_rtscope_n2a(), and snprintf_pkt().

7.26.3.2 const char* name

Definition at line 35 of file ll_proto.c.

Referenced by __get_hz(), ll_proto_a2n(), and ll_type_n2a().

7.27 src/libnetlink/ll_types.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <syslog.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <linux/netdevice.h>
#include <linux/if_arp.h>
#include <linux/sockios.h>
#include "rt_names.h"
```

Macros

- #define __PF(f, n) { ARPHRD_##f, #n },

Functions

- const char * ll_type_n2a (int type, char *buf, int len)

7.27.1 Macro Definition Documentation

7.27.1.1 #define __PF(*f*, *n*) { ARPHRD_##f, #n },

7.27.2 Function Documentation

7.27.2.1 const char* ll_type_n2a (int *type*, char * *buf*, int *len*)

Definition at line 30 of file ll_types.c.

References __PF, and name.

```
                                                    {
#define __PF(f,n) { ARPHRD_##f, #n },
    static const struct {
        int type;
        const char *name;
    } arphrd_names[] = {
        { 0, "generic" },
        __PF(ETHER,ether)
        __PF(EETHER,eether)
        __PF(AX25,ax25)
        __PF(PRONET,pronet)
        __PF(CHAOS,chaos)
        __PF(IEEE802,ieee802)
        __PF(ARCNET,arcnet)
        __PF(APPLETLK,atalk)
        __PF(DLCI,dlci)
        __PF(ATM,atm)
        __PF(METRICOM,metricom)
        __PF(IEEE1394,ieee1394)
        __PF(INFINIBAND,infiniband)
        __PF(SLIP,slip)
        __PF(CSLIP,cslip)
        __PF(SLIP6,slip6)
        __PF(CSLIP6,cslip6)
        __PF(RSRVD,rsrvd)
        __PF(ADAPT,adapt)
        __PF(ROSE,rose)
        __PF(X25,x25)
        __PF(HWX25,hwx25)
        __PF(CAN,can)
        __PF(PPP,ppp)
        __PF(HDLC,hdlc)
        __PF(LAPB,lapb)
        __PF(DDCMP,ddcmp)
        __PF(RAWHDLC,rawhdlc)
        __PF(TUNNEL,ipip)
        __PF(TUNNEL6,tunnel6)
        __PF(FRAD,frad)
        __PF(SKIP,skip)
        __PF(LOOPBACK,loopback)
        __PF(LOCALTLK,ltalk)
        __PF(FDDI,fddi)
        __PF(BIF,bif)
        __PF(SIT,sit)
        __PF(IPDDP,ip/ddp)
        __PF(IPGRE,gre)
        __PF(PIMREG,pimreg)
        __PF(HIPPI,hippi)
        __PF(ASH,ash)
        __PF(ECONET,econet)
        __PF(IRDA,irda)
        __PF(FCPP,fcpp)
        __PF(FCAL,fcal)
        __PF(FCPL,fcpl)
        __PF(FCFABRIC,fcfb0)
        __PF(FCFABRIC+1,fcfb1)
        __PF(FCFABRIC+2,fcfb2)
        __PF(FCFABRIC+3,fcfb3)
        __PF(FCFABRIC+4,fcfb4)
        __PF(FCFABRIC+5,fcfb5)
        __PF(FCFABRIC+6,fcfb6)
        __PF(FCFABRIC+7,fcfb7)
        __PF(FCFABRIC+8,fcfb8)
        __PF(FCFABRIC+9,fcfb9)
        __PF(FCFABRIC+10,fcfb10)
        __PF(FCFABRIC+11,fcfb11)
        __PF(FCFABRIC+12,fcfb12)
        __PF(IEEE802_TR,tr)
        __PF(IEEE80211,ieee802.11)
        __PF(IEEE80211_PRISM,ieee802.11/prism)
        __PF(IEEE80211_RADIOTAP,ieee802.11/radiotap)
        __PF(IEEE802154, ieee802.15.4)
        __PF(PHONET, phonet)
        __PF(PHONET_PIPE, phonet_pipe)
        __PF(CAIF, caif)
```

```
        __PF(NONE, none)
        __PF(VOID,void)
    };
#undef __PF

    int i;
    for (i=0; i<sizeof(arphrd_names)/sizeof(arphrd_names[0]); i++) {
        if (arphrd_names[i].type == type) {
            return arphrd_names[i].name;
        }
    }
    snprintf(buf, len, "[%d]", type);
    return buf;
}
```

7.28 src/libnetlink/rt_names.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <syslog.h>
#include <fcntl.h>
#include <string.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <asm/types.h>
#include <linux/rtnetlink.h>
#include "rt_names.h"
```

Data Structures

- struct rtnl_hash_entry

Macros

- #define CONFDIR "/etc/iproute2"

Functions

- char * rtnl_rtprot_n2a (int id, char *buf, int len)
- int rtnl_rtprot_a2n (__u32 *id, char *arg)
- char * rtnl_rtscope_n2a (int id, char *buf, int len)
- int rtnl_rtscope_a2n (__u32 *id, char *arg)
- char * rtnl_rtrealm_n2a (int id, char *buf, int len)
- int rtnl_rtrealm_a2n (__u32 *id, char *arg)
- char * rtnl_rttable_n2a (__u32 id, char *buf, int len)
- int rtnl_rttable_a2n (__u32 *id, char *arg)
- char * rtnl_dsfield_n2a (int id, char *buf, int len)
- int rtnl_dsfield_a2n (__u32 *id, char *arg)
- int rtnl_group_a2n (int *id, char *arg)

7.28.1 Macro Definition Documentation

7.28.1.1 #define CONFDIR "/etc/iproute2"

Definition at line 27 of file rt_names.c.

7.28.2 Function Documentation

7.28.2.1 int rtnl_dsfield_a2n (__u32 * id, char * arg)

Definition at line 436 of file rt_names.c.

```
                                                {
    static char *cache = NULL;
    static unsigned long res;
    char *end;
    int i;

    if (cache && strcmp(cache, arg) == 0) {
        *id = res;
        return 0;
    }

    if (!rtnl_rtdsfield_init) {
        rtnl_rtdsfield_initialize();
    }

    for (i=0; i<256; i++) {
        if (rtnl_rtdsfield_tab[i] &&
                strcmp(rtnl_rtdsfield_tab[i], arg) == 0) {
            cache = rtnl_rtdsfield_tab[i];
            res = i;
            *id = res;
            return 0;
        }
    }

    res = strtoul(arg, &end, 16);
    if (!end || end == arg || *end || res > 255) {
        return -1;
    }
    *id = res;
    return 0;
}
```

7.28.2.2 char* rtnl_dsfield_n2a (int id, char * buf, int len)

Definition at line 418 of file rt_names.c.

References id.

```
                                                    {
    if (id<0 || id>=256) {
        snprintf(buf, len, "%d", id);
        return buf;
    }
    if (!rtnl_rtdsfield_tab[id]) {
        if (!rtnl_rtdsfield_init) {
            rtnl_rtdsfield_initialize();
        }
    }
    if (rtnl_rtdsfield_tab[id]) {
        return rtnl_rtdsfield_tab[id];
    }
    snprintf(buf, len, "0x%02x", id);
    return buf;
}
```

7.28.2.3 int rtnl_group_a2n (int * id, char * arg)

Definition at line 484 of file rt_names.c.

References rtnl_hash_entry::id, rtnl_hash_entry::name, and rtnl_hash_entry::next.

```
                                                {
    static char *cache = NULL;
    static unsigned long res;
    struct rtnl_hash_entry *entry;
    char *end;
```

```
    int i;

    if (cache && strcmp(cache, arg) == 0) {
        *id = res;
        return 0;
    }

    if (!rtnl_group_init) {
        rtnl_group_initialize();
    }

    for (i=0; i<256; i++) {
        entry = rtnl_group_hash[i];
        while (entry && strcmp(entry->name, arg)) {
            entry = entry->next;
        }
        if (entry) {
            cache = entry->name;
            res = entry->id;
            *id = res;
            return 0;
        }
    }

    i = strtol(arg, &end, 0);
    if (!end || end == arg || *end || i < 0) {
        return -1;
    }
    *id = i;
    return 0;
}
```

7.28.2.4 int rtnl_rtprot_a2n (__u32 * *id*, char * *arg*)

Definition at line 162 of file rt_names.c.

```
                                        {
    static char *cache = NULL;
    static unsigned long res;
    char *end;
    int i;

    if (cache && strcmp(cache, arg) == 0) {
        *id = res;
        return 0;
    }

    if (!rtnl_rtprot_init) {
        rtnl_rtprot_initialize();
    }

    for (i=0; i<256; i++) {
        if (rtnl_rtprot_tab[i] &&
                strcmp(rtnl_rtprot_tab[i], arg) == 0) {
            cache = rtnl_rtprot_tab[i];
            res = i;
            *id = res;
            return 0;
        }
    }

    res = strtoul(arg, &end, 0);
    if (!end || end == arg || *end || res > 255) {
        return -1;
    }
    *id = res;
    return 0;
}
```

7.28.2.5 char* rtnl_rtprot_n2a (int *id*, char * *buf*, int *len*)

Definition at line 145 of file rt_names.c.

References id.

```
                                                {
    if (id<0 || id>=256) {
```

```
        snprintf(buf, len, "%d", id);
        return buf;
    }
    if (!rtnl_rtprot_tab[id]) {
        if (!rtnl_rtprot_init) {
            rtnl_rtprot_initialize();
        }
    }
    if (rtnl_rtprot_tab[id]) {
        return rtnl_rtprot_tab[id];
    }
    snprintf(buf, len, "%d", id);
    return buf;
}
```

7.28.2.6 int rtnl_rtrealm_a2n (__u32 * *id*, char * *arg*)

Definition at line 295 of file rt_names.c.

```
                                        {
    static char *cache = NULL;
    static unsigned long res;
    char *end;
    int i;

    if (cache && strcmp(cache, arg) == 0) {
        *id = res;
        return 0;
    }

    if (!rtnl_rtrealm_init) {
        rtnl_rtrealm_initialize();
    }

    for (i=0; i<256; i++) {
        if (rtnl_rtrealm_tab[i] &&
                strcmp(rtnl_rtrealm_tab[i], arg) == 0) {
            cache = rtnl_rtrealm_tab[i];
            res = i;
            *id = res;
            return 0;
        }
    }

    res = strtoul(arg, &end, 0);
    if (!end || end == arg || *end || res > 255) {
        return -1;
    }
    *id = res;
    return 0;
}
```

7.28.2.7 char* rtnl_rtrealm_n2a (int *id*, char * *buf*, int *len*)

Definition at line 277 of file rt_names.c.

References id.

```
                                                    {
    if (id<0 || id>=256) {
        snprintf(buf, len, "%d", id);
        return buf;
    }
    if (!rtnl_rtrealm_tab[id]) {
        if (!rtnl_rtrealm_init) {
            rtnl_rtrealm_initialize();
        }
    }
    if (rtnl_rtrealm_tab[id]) {
        return rtnl_rtrealm_tab[id];
    }
    snprintf(buf, len, "%d", id);
    return buf;
}
```

7.28.2.8 int rtnl_rtscope_a2n (__u32 * id, char * arg)

Definition at line 230 of file rt_names.c.

```
                                        {
    static char *cache = NULL;
    static unsigned long res;
    char *end;
    int i;

    if (cache && strcmp(cache, arg) == 0) {
        *id = res;
        return 0;
    }

    if (!rtnl_rtscope_init) {
        rtnl_rtscope_initialize();
    }

    for (i=0; i<256; i++) {
        if (rtnl_rtscope_tab[i] &&
                strcmp(rtnl_rtscope_tab[i], arg) == 0) {
            cache = rtnl_rtscope_tab[i];
            res = i;
            *id = res;
            return 0;
        }
    }

    res = strtoul(arg, &end, 0);
    if (!end || end == arg || *end || res > 255) {
        return -1;
    }
    *id = res;
    return 0;
}
```

7.28.2.9 char* rtnl_rtscope_n2a (int id, char * buf, int len)

Definition at line 213 of file rt_names.c.

References id.

```
                                                {
    if (id<0 || id>=256) {
        snprintf(buf, len, "%d", id);
        return buf;
    }
    if (!rtnl_rtscope_tab[id]) {
        if (!rtnl_rtscope_init) {
            rtnl_rtscope_initialize();
        }
    }
    if (rtnl_rtscope_tab[id]) {
        return rtnl_rtscope_tab[id];
    }
    snprintf(buf, len, "%d", id);
    return buf;
}
```

7.28.2.10 int rtnl_rttable_a2n (__u32 * id, char * arg)

Definition at line 368 of file rt_names.c.

References rtnl_hash_entry::id, rtnl_hash_entry::name, and rtnl_hash_entry::next.

```
                                        {
    static char *cache = NULL;
    static unsigned long res;
    struct rtnl_hash_entry *entry;
    char *end;
    __u32 i;

    if (cache && strcmp(cache, arg) == 0) {
```

```
        *id = res;
        return 0;
    }

    if (!rtnl_rttable_init) {
        rtnl_rttable_initialize();
    }

    for (i=0; i<256; i++) {
        entry = rtnl_rttable_hash[i];
        while (entry && strcmp(entry->name, arg)) {
            entry = entry->next;
        }
        if (entry) {
            cache = entry->name;
            res = entry->id;
            *id = res;
            return 0;
        }
    }

    i = strtoul(arg, &end, 0);
    if (!end || end == arg || *end || i > RT_TABLE_MAX) {
        return -1;
    }
    *id = i;
    return 0;
}
```

7.28.2.11 char* rtnl_rttable_n2a (__u32 *id*, char * *buf*, int *len*)

Definition at line 347 of file rt_names.c.

References rtnl_hash_entry::id, rtnl_hash_entry::name, and rtnl_hash_entry::next.

```
                                                    {
    struct rtnl_hash_entry *entry;

    if (id > RT_TABLE_MAX) {
        snprintf(buf, len, "%u", id);
        return buf;
    }
    if (!rtnl_rttable_init) {
        rtnl_rttable_initialize();
    }
    entry = rtnl_rttable_hash[id & 255];
    while (entry && entry->id != id) {
        entry = entry->next;
    }
    if (entry) {
        return entry->name;
    }
    snprintf(buf, len, "%u", id);
    return buf;
}
```

7.29 src/libnetlink/utils.c File Reference

```
#include <stdio.h>
```

```
#include <stdlib.h>
#include <unistd.h>
#include <syslog.h>
#include <fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <resolv.h>
#include <asm/types.h>
#include <linux/pkt_sched.h>
#include <time.h>
#include <sys/time.h>
#include <errno.h>
#include "utils.h"
```

Functions

- int get_integer (int *val, const char *arg, int base)
- int mask2bits (__u32 netmask)
- int get_unsigned (unsigned *val, const char *arg, int base)
- int get_time_rtt (unsigned *val, const char *arg, int *raw)
- int get_u64 (__u64 *val, const char *arg, int base)
- int get_u32 (__u32 *val, const char *arg, int base)
- int get_u16 (__u16 *val, const char *arg, int base)
- int get_u8 (__u8 *val, const char *arg, int base)
- int get_s32 (__s32 *val, const char *arg, int base)
- int get_s16 (__s16 *val, const char *arg, int base)
- int get_s8 (__s8 *val, const char *arg, int base)
- int get_addr_1 (inet_prefix *addr, const char *name, int family)
- int get_prefix_1 (inet_prefix *dst, char *arg, int family)
- int get_addr (inet_prefix *dst, const char *arg, int family)
- int get_prefix (inet_prefix *dst, char *arg, int family)
- __u32 get_addr32 (const char *name)
- void incomplete_command (void)
- void missarg (const char *key)
- void invarg (const char *msg, const char *arg)
- void duparg (const char *key, const char *arg)
- void duparg2 (const char *key, const char *arg)
- int matches (const char *cmd, const char *pattern)
- int inet_addr_match (const inet_prefix *a, const inet_prefix *b, int bits)
- int __get_hz (void)
- int __get_user_hz (void)
- const char * rt_addr_n2a (int af, int len, const void *addr, char *buf, int buflen)
- const char * format_host (int af, int len, const void *addr, char *buf, int buflen)
- char * hexstring_n2a (const __u8 *str, int len, char *buf, int blen)
- __u8 * hexstring_a2n (const char *str, __u8 *buf, int blen)
- int print_timestamp (FILE *fp)
- ssize_t getcmdline (char **linep, size_t *lenp, FILE *in)
- int makeargs (char *line, char *argv[], int maxargs)

Variables

- int __iproute2_hz_internal
- int __iproute2_user_hz_internal
- int cmdlineno

7.29.1 Function Documentation

7.29.1.1 int __get_hz (void)

Definition at line 502 of file utils.c.

References name.

```
                {
    char name[1024];
    int hz = 0;
    FILE *fp;

    if (getenv("HZ")) {
        return atoi(getenv("HZ")) ? : HZ;
    }

    if (getenv("PROC_NET_PSCHED")) {
        snprintf(name, sizeof(name)-1, "%s", getenv("PROC_NET_PSCHED"));
    } else
        if (getenv("PROC_ROOT")) {
            snprintf(name, sizeof(name)-1, "%s/net/psched", getenv("PROC_ROOT")
      );
        } else {
            strcpy(name, "/proc/net/psched");
        }
    fp = fopen(name, "r");

    if (fp) {
        unsigned nom, denom;
        if (fscanf(fp, "%*08x%*08x%08x%08x", &nom, &denom) == 2)
            if (nom == 1000000) {
                hz = denom;
            }
        fclose(fp);
    }
    if (hz) {
        return hz;
    }
    return HZ;
}
```

7.29.1.2 int __get_user_hz (void)

Definition at line 537 of file utils.c.

```
                    {
    return sysconf(_SC_CLK_TCK);
}
```

7.29.1.3 void duparg (const char * *key*, const char * *arg*)

Definition at line 453 of file utils.c.

```
                                        {
    fprintf(stderr, "Error: duplicate \"%s\": \"%s\" is the second value.\n",
      key, arg);
    exit(-1);
}
```

7.29.1.4 void duparg2 (const char * *key*, const char * *arg*)

Definition at line 458 of file utils.c.

```
                                                {
    fprintf(stderr, "Error: either \"%s\" is duplicate, or \"%s\" is a garbage.
      \n", key, arg);
    exit(-1);
}
```

7.29.1.5 const char* format_host (int *af*, int *len*, const void * *addr*, char * *buf*, int *buflen*)

Definition at line 616 of file utils.c.

References AF_DECnet, resolve_hosts, and rt_addr_n2a().

```
                                                        {
#ifdef RESOLVE_HOSTNAMES
    if (resolve_hosts) {
        const char *n;

        if (len <= 0) {
            switch (af) {
                case AF_INET:
                    len = 4;
                    break;
                case AF_INET6:
                    len = 16;
                    break;
                case AF_IPX:
                    len = 10;
                    break;
#ifdef AF_DECnet
                    /* I see no reasons why gethostbyname
                       may not work for DECnet */
                case AF_DECnet:
                    len = 2;
                    break;
#endif
                default
                        :
                    ;
            }
        }
        if (len > 0 &&
                (n = resolve_address(addr, len, af)) != NULL) {
            return n;
        }
    }
#endif
    return rt_addr_n2a(af, len, addr, buf, buflen);
}
```

7.29.1.6 int get_addr (inet_prefix * *dst*, const char * *arg*, int *family*)

Definition at line 405 of file utils.c.

References get_addr_1().

```
                                                                {
    if (family == AF_PACKET) {
        fprintf(stderr, "Error: \"%s\" may be inet address, but it is not
       allowed in this context.\n", arg);
        exit(1);
    }
    if (get_addr_1(dst, arg, family)) {
        fprintf(stderr, "Error: an inet address is expected rather than \"%s\".
      \n", arg);
        exit(1);
    }
    return 0;
}
```

7.29.1.7 __u32 get_addr32 (const char * *name*)

Definition at line 429 of file utils.c.

References inet_prefix::data, and get_addr_1().

```
                                    {
    inet_prefix addr;
    if (get_addr_1(&addr, name, AF_INET)) {
        fprintf(stderr, "Error: an IP address is expected rather than \"%s\"\n"
      , name);
        exit(1);
    }
    return addr.data[0];
}
```

7.29.1.8 int get_addr_1 (inet_prefix * *addr,* const char * *name,* int *family*)

Definition at line 296 of file utils.c.

References dn_naddr::a_addr, AF_DECnet, inet_prefix::bitlen, inet_prefix::bytelen, inet_prefix::data, dnet_pton(), and inet_prefix::family.

Referenced by get_addr(), get_addr32(), get_prefix_1(), and ll_addr_a2n().

```
                                                                    {
    memset(addr, 0, sizeof(*addr));

    if (strcmp(name, "default") == 0 ||
            strcmp(name, "all") == 0 ||
            strcmp(name, "any") == 0) {
        if (family == AF_DECnet) {
            return -1;
        }
        addr->family = family;
        addr->bytelen = (family == AF_INET6 ? 16 : 4);
        addr->bitlen = -1;
        return 0;
    }

    if (strchr(name, ':')) {
        addr->family = AF_INET6;
        if (family != AF_UNSPEC && family != AF_INET6) {
            return -1;
        }
        if (inet_pton(AF_INET6, name, addr->data) <= 0) {
            return -1;
        }
        addr->bytelen = 16;
        addr->bitlen = -1;
        return 0;
    }

    if (family == AF_DECnet) {
        struct dn_naddr dna;
        addr->family = AF_DECnet;
        if (dnet_pton(AF_DECnet, name, &dna) <= 0) {
            return -1;
        }
        memcpy(addr->data, dna.a_addr, 2);
        addr->bytelen = 2;
        addr->bitlen = -1;
        return 0;
    }

    addr->family = AF_INET;
    if (family != AF_UNSPEC && family != AF_INET) {
        return -1;
    }

    if (get_addr_ipv4((__u8 *)addr->data, name) <= 0) {
        return -1;
    }

    addr->bytelen = 4;
    addr->bitlen = -1;
    return 0;
}
```

7.29.1.9 int get_integer (int * *val*, const char * *arg*, int *base*)

Definition at line 33 of file utils.c.

```
{
    long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtol(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res > INT_MAX || res < INT_MIN) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.29.1.10 int get_prefix (inet_prefix * *dst*, char * *arg*, int *family*)

Definition at line 417 of file utils.c.

References get_prefix_1().

Referenced by set_interface_ipaddr().

```
{
    if (family == AF_PACKET) {
        fprintf(stderr, "Error: \"%s\" may be inet prefix, but it is not
       allowed in this context.\n", arg);
        exit(1);
    }
    if (get_prefix_1(dst, arg, family)) {
        fprintf(stderr, "Error: an inet prefix is expected rather than \"%s\".
      \n", arg);
        exit(1);
    }
    return 0;
}
```

7.29.1.11 int get_prefix_1 (inet_prefix * *dst*, char * *arg*, int *family*)

Definition at line 350 of file utils.c.

References AF_DECnet, inet_prefix::bitlen, inet_prefix::bytelen, inet_prefix::family, inet_prefix::flags, get_addr_1(), and PREFIXLEN_SPECIFIED.

Referenced by get_prefix().

```
{
    int err;
    unsigned plen;
    char *slash;

    memset(dst, 0, sizeof(*dst));

    if (strcmp(arg, "default") == 0 ||
            strcmp(arg, "any") == 0 ||
            strcmp(arg, "all") == 0) {
        if (family == AF_DECnet) {
            return -1;
        }
        dst->family = family;
        dst->bytelen = 0;
        dst->bitlen = 0;
        return 0;
    }

    slash = strchr(arg, '/');
    if (slash) {
        *slash = 0;
    }
```

```
    err = get_addr_1(dst, arg, family);
    if (err == 0) {
        switch(dst->family) {
            case AF_INET6:
                dst->bitlen = 128;
                break;
            case AF_DECnet:
                dst->bitlen = 16;
                break;
            default
                    :
            case AF_INET:
                dst->bitlen = 32;
        }
        if (slash) {
            if (get_netmask(&plen, slash+1, 0)
                    || plen > dst->bitlen) {
                err = -1;
                goto done;
            }
            dst->flags |= PREFIXLEN_SPECIFIED;
            dst->bitlen = plen;
        }
    }
done:
    if (slash) {
        *slash = '/';
    }
    return err;
}
```

7.29.1.12 int get_s16 (__s16 * *val*, const char * *arg*, int *base*)

Definition at line 232 of file utils.c.

```
                                                    {
    long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtol(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res > 0x7FFF || res < -0x8000) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.29.1.13 int get_s32 (__s32 * *val*, const char * *arg*, int *base*)

Definition at line 213 of file utils.c.

```
                                                    {
    long res;
    char *ptr;

    errno = 0;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtol(arg, &ptr, base);
    if (ptr == arg || *ptr ||
            ((res ==  LONG_MIN || res == LONG_MAX) && errno == ERANGE) ||
            res > INT32_MAX || res < INT32_MIN) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.29.1.14 int get_s8 (__s8 * *val*, const char * *arg*, int *base*)

Definition at line 247 of file utils.c.

```
                                                    {
    long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtol(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res > 0x7F || res < -0x80) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.29.1.15 int get_time_rtt (unsigned * *val*, const char * *arg*, int * *raw*)

Definition at line 106 of file utils.c.

```
                                                                  {
    double t;
    unsigned long res;
    char *p;

    if (strchr(arg,'.') != NULL) {
        t = strtod(arg,&p);
        if (t < 0.0) {
            return -1;
        }
    } else {
        res = strtoul(arg, &p, 0);
        if (res > UINT_MAX) {
            return -1;
        }
        t = (double)res;
    }
    if (p == arg) {
        return -1;
    }
    *raw = 1;

    if (*p) {
        *raw = 0;
        if (strcasecmp(p, "s") == 0 || strcasecmp(p, "sec")==0 ||
                strcasecmp(p, "secs")==0) {
            t *= 1000;
        } else
            if (strcasecmp(p, "ms") == 0 || strcasecmp(p, "msec")==0 ||
                    strcasecmp(p, "msecs") == 0) {
                t *= 1.0;    /* allow suffix, do nothing */
            } else {
                return -1;
            }
    }

    /* emulate ceil() without having to bring-in -lm and always be >= 1 */

    *val = t;
    if (*val < t) {
        *val += 1;
    }

    return 0;

}
```

7.29.1.16 int get_u16 (__u16 * *val*, const char * *arg*, int *base*)

Definition at line 183 of file utils.c.

Referenced by ll_proto_a2n().

```
                                                {
    unsigned long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtoul(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res > 0xFFFF) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.29.1.17 int get_u32 (__u32 * *val*, const char * *arg*, int *base*)

Definition at line 168 of file utils.c.

```
                                                {
    unsigned long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtoul(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res > 0xFFFFFFFFUL) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.29.1.18 int get_u64 (__u64 * *val*, const char * *arg*, int *base*)

Definition at line 153 of file utils.c.

```
                                                {
    unsigned long long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtoull(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res == 0xFFFFFFFFULL) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.29.1.19 int get_u8 (__u8 * *val*, const char * *arg*, int *base*)

Definition at line 198 of file utils.c.

Referenced by inet_proto_a2n().

```
                                                {
    unsigned long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtoul(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res > 0xFF) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.29.1.20 int get_unsigned (unsigned * *val*, const char * *arg*, int *base*)

Definition at line 84 of file utils.c.

```
{
    unsigned long res;
    char *ptr;

    if (!arg || !*arg) {
        return -1;
    }
    res = strtoul(arg, &ptr, base);
    if (!ptr || ptr == arg || *ptr || res > UINT_MAX) {
        return -1;
    }
    *val = res;
    return 0;
}
```

7.29.1.21 ssize_t getcmdline (char ** *linep*, size_t * *lenp*, FILE * *in*)

Definition at line 733 of file utils.c.

References cmdlineno, and realloc.

```
{
    ssize_t cc;
    char *cp;

    if ((cc = getline(linep, lenp, in)) < 0) {
        return cc;    /* eof or error */
    }
    ++cmdlineno;

    cp = strchr(*linep, '#');
    if (cp) {
        *cp = '\0';
    }

    while ((cp = strstr(*linep, "\\\n")) != NULL) {
        char *line1 = NULL;
        size_t len1 = 0;
        ssize_t cc1;

        if ((cc1 = getline(&line1, &len1, in)) < 0) {
            fprintf(stderr, "Missing continuation line\n");
            return cc1;
        }

        ++cmdlineno;
        *cp = 0;

        cp = strchr(line1, '#');
        if (cp) {
            *cp = '\0';
        }

        *lenp = strlen(*linep) + strlen(line1) + 1;
        *linep = realloc(*linep, *lenp);
        if (!*linep) {
            fprintf(stderr, "Out of memory\n");
            *lenp = 0;
            return -1;
        }
        cc += cc1 - 2;
        strcat(*linep, line1);
        free(line1);
    }
    return cc;
}
```

7.29.1.22 __u8* hexstring_a2n (const char * *str*, __u8 * *buf*, int *blen*)

Definition at line 674 of file utils.c.

```
                                                                {
    int cnt = 0;

    for (;;) {
        unsigned acc;
        char ch;

        acc = 0;

        while ((ch = *str) != ':' && ch != 0) {
            if (ch >= '0' && ch <= '9') {
                ch -= '0';
            } else
                if (ch >= 'a' && ch <= 'f') {
                    ch -= 'a'-10;
                } else
                    if (ch >= 'A' && ch <= 'F') {
                        ch -= 'A'-10;
                    } else {
                        return NULL;
                    }
            acc = (acc<<4) + ch;
            str++;
        }

        if (acc > 255) {
            return NULL;
        }
        if (cnt < blen) {
            buf[cnt] = acc;
            cnt++;
        }
        if (ch == 0) {
            break;
        }
        ++str;
    }
    if (cnt < blen) {
        memset(buf+cnt, 0, blen-cnt);
    }
    return buf;
}
```

7.29.1.23 char* hexstring_n2a (const __u8 * *str*, int *len*, char * *buf*, int *blen*)

Definition at line 655 of file utils.c.

```
                                                                    {
    char *ptr = buf;
    int i;

    for (i=0; i<len; i++) {
        if (blen < 3) {
            break;
        }
        sprintf(ptr, "%02x", str[i]);
        ptr += 2;
        blen -= 2;
        if (i != len-1 && blen > 1) {
            *ptr++ = ':';
            blen--;
        }
    }
    return buf;
}
```

7.29.1.24 void incomplete_command (void)

Definition at line 438 of file utils.c.

```
                            {
    fprintf(stderr, "Command line is not complete. Try option \"help\"\n");
    exit(-1);
}
```

7.29.1.25 int inet_addr_match (const inet_prefix * a, const inet_prefix * b, int bits)

Definition at line 471 of file utils.c.

References inet_prefix::data.

```
                                                                                {
    const __u32 *a1 = a->data;
    const __u32 *a2 = b->data;
    int words = bits >> 0x05;

    bits &= 0x1f;

    if (words)
        if (memcmp(a1, a2, words << 2)) {
            return -1;
        }

    if (bits) {
        __u32 w1, w2;
        __u32 mask;

        w1 = a1[words];
        w2 = a2[words];

        mask = htonl((0xffffffff) << (0x20 - bits));

        if ((w1 ^ w2) & mask) {
            return 1;
        }
    }

    return 0;
}
```

7.29.1.26 void invarg (const char * msg, const char * arg)

Definition at line 448 of file utils.c.

```
                                                {
    fprintf(stderr, "Error: argument \"%s\" is wrong: %s\n", arg, msg);
    exit(-1);
}
```

7.29.1.27 int makeargs (char * line, char * argv[], int maxargs)

Definition at line 780 of file utils.c.

```
                                                                {
    static const char ws[] = " \t\r\n";
    char *cp;
    int argc = 0;

    for (cp = strtok(line, ws); cp; cp = strtok(NULL, ws)) {
        if (argc >= (maxargs - 1)) {
            fprintf(stderr, "Too many arguments to command\n");
            exit(1);
        }
        argv[argc++] = cp;
    }
    argv[argc] = NULL;

    return argc;
}
```

7.29.1.28 int mask2bits (__u32 netmask)

Definition at line 48 of file utils.c.

```
                                {
    unsigned bits = 0;
    __u32 mask = ntohl(netmask);
    __u32 host = ~mask;

    /* a valid netmask must be 2^n - 1 */
    if ((host & (host + 1)) != 0) {
        return -1;
    }

    for (; mask; mask <<= 1) {
        ++bits;
    }
    return bits;
}
```

7.29.1.29 int matches (const char * *cmd,* const char * *pattern*)

Definition at line 463 of file utils.c.

```
                                                        {
    int len = strlen(cmd);
    if (len > strlen(pattern)) {
        return -1;
    }
    return memcmp(pattern, cmd, len);
}
```

7.29.1.30 void missarg (const char * *key*)

Definition at line 443 of file utils.c.

```
                                {
    fprintf(stderr, "Error: argument \"%s\" is required\n", key);
    exit(-1);
}
```

7.29.1.31 int print_timestamp (FILE * *fp*)

Definition at line 717 of file utils.c.

```
                                {
    struct timeval tv;
    char *tstr;

    memset(&tv, 0, sizeof(tv));
    gettimeofday(&tv, NULL);

    tstr = asctime(localtime(&tv.tv_sec));
    tstr[strlen(tstr)-1] = 0;
    fprintf(fp, "Timestamp: %s %lu usec\n", tstr, tv.tv_usec);
    return 0;
}
```

7.29.1.32 const char* rt_addr_n2a (int *af,* int *len,* const void * *addr,* char * *buf,* int *buflen*)

Definition at line 541 of file utils.c.

References dn_naddr::a_addr, AF_DECnet, dnet_ntop(), and ipx_ntop().

Referenced by format_host().

```
        {
    switch (af) {
        case AF_INET:
```

```
        case AF_INET6:
            return inet_ntop(af, addr, buf, buflen);
        case AF_IPX:
            return ipx_ntop(af, addr, buf, buflen);
        case AF_DECnet: {
            struct dn_naddr dna = { 2, { 0, 0, }};
            memcpy(dna.a_addr, addr, 2);
            return dnet_ntop(af, &dna, buf, buflen);
        }
        default
                :
            return "???";
    }
}
```

7.29.2 Variable Documentation

7.29.2.1 int __iproute2_hz_internal

Definition at line 500 of file utils.c.

7.29.2.2 int __iproute2_user_hz_internal

Definition at line 535 of file utils.c.

7.29.2.3 int cmdlineno

Definition at line 730 of file utils.c.

Referenced by getcmdline().

7.30 src/libxml2.c File Reference

XML Interface.

```
#include <string.h>
#include <stdint.h>
#include <libxml/tree.h>
#include <libxml/parser.h>
#include <libxml/xpath.h>
#include <libxml/xpathInternals.h>
#include "include/priv_xml.h"
#include "include/dtsapp.h"
```

Data Structures

- struct xml_node_iter
- struct xml_search

Functions

- void free_buffer (void *data)
- int node_hash (const void *data, int key)
- int attr_hash (const void *data, int key)
- struct xml_doc * xml_loaddoc (const char *docfile, int validate)
- struct xml_doc * xml_loadbuf (const uint8_t *buffer, uint32_t len, int validate)

- struct xml_node * xml_nodetohash (struct xml_doc *xmldoc, xmlNodePtr node, const char *attrkey)
- struct xml_node * xml_gethash (struct xml_search *xpsearch, int i, const char *attrkey)
- struct xml_node * xml_getrootnode (struct xml_doc *xmldoc)
- struct xml_node * xml_getfirstnode (struct xml_search *xpsearch, void **iter)
- struct xml_node * xml_getnextnode (void *iter)
- struct bucket_list * xml_getnodes (struct xml_search *xpsearch)
- struct bucket_list * xml_setnodes (struct xml_search *xpsearch, const char *attrkey)
- struct xml_search * xml_xpath (struct xml_doc *xmldata, const char *xpath, const char *attrkey)
- int xml_nodecount (struct xml_search *xsearch)
- struct xml_node * xml_getnode (struct xml_search *xsearch, const char *key)
- const char * xml_getattr (struct xml_node *xnode, const char *attr)
- const char * xml_getrootname (struct xml_doc *xmldoc)
- void xml_modify (struct xml_doc *xmldoc, struct xml_node *xnode, const char *value)
- void xml_setattr (struct xml_doc *xmldoc, struct xml_node *xnode, const char *name, const char *value)
- void xml_createpath (struct xml_doc *xmldoc, const char *xpath)
- void xml_appendnode (struct xml_doc *xmldoc, const char *xpath, struct xml_node *child)
- struct xml_node * xml_addnode (struct xml_doc *xmldoc, const char *xpath, const char *name, const char *value, const char *attrkey, const char *keyval)
- void xml_unlink (struct xml_node *xnode)
- void xml_delete (struct xml_node *xnode)
- char * xml_getbuffer (void *buffer)
- void * xml_doctobuffer (struct xml_doc *xmldoc)
- void xml_init ()
- void xml_close ()
- void xml_savefile (struct xml_doc *xmldoc, const char *file, int format, int compress)
- void xml_modify2 (struct xml_search *xpsearch, struct xml_node *xnode, const char *value)

Variables

- int xmlLoadExtwDtdDefaultValue

7.30.1 Detailed Description

XML Interface.

Definition in file libxml2.c.

7.31 src/libxslt.c File Reference

XSLT Interface.

```
#include <stdint.h>
#include <string.h>
#include <libxslt/xsltutils.h>
#include <libxslt/transform.h>
#include "include/dtsapp.h"
#include "include/priv_xml.h"
```

Data Structures

- struct xslt_doc
- struct xslt_param

Functions

- void free_xsltdoc (void *data)
- void free_parser (void *data)
- int xslt_hash (const void *data, int key)
- struct xslt_doc * xslt_open (const char *xsltfile)
- void free_param (void *data)
- void xslt_addparam (struct xslt_doc *xsltdoc, const char *param, const char *value)
- void xslt_clearparam (struct xslt_doc *xsltdoc)
- void xslt_apply (struct xml_doc *xmldoc, struct xslt_doc *xsltdoc, const char *filename, int comp)
- void * xslt_apply_buffer (struct xml_doc *xmldoc, struct xslt_doc *xsltdoc)
- void xslt_init ()
- void xslt_close ()

7.31.1 Detailed Description

XSLT Interface.

Definition in file libxslt.c.

7.32 src/lookup3.c File Reference

by Bob Jenkins, May 2006, Public Domain.

```
#include <stdio.h>
#include <time.h>
#include <stdint.h>
#include <sys/param.h>
```

Macros

- #define HASH_LITTLE_ENDIAN 0
- #define HASH_BIG_ENDIAN 0
- #define hashsize(n) ((uint32_t)1<<(n))
- #define hashmask(n) (hashsize(n)-1)
- #define rot(x, k) (((x)<<(k)) | ((x)>>(32-(k))))
- #define mix(a, b, c)

  mix 3 32-bit values reversibly
- #define final(a, b, c)

  final mixing of 3 32-bit values (a,b,c) into c

Functions

- uint32_t hashword (const uint32_t *k, size_t length, uint32_t initval)

  hash a variable-length key into a 32-bit value (Big Endian)
- void hashword2 (const uint32_t *k, size_t length, uint32_t *pc, uint32_t *pb)

  same as hashword(), but take two seeds and return two 32-bit values
- uint32_t hashlittle (const void *key, size_t length, uint32_t initval)

  hash a variable-length key into a 32-bit value (Little Endian)
- void hashlittle2 (const void *key, size_t length, uint32_t *pc, uint32_t *pb)

  return 2 32-bit hash values.
- uint32_t hashbig (const void *key, size_t length, uint32_t initval)

  This is the same as hashword() on big-endian machines.

7.32.1 Detailed Description

by Bob Jenkins, May 2006, Public Domain.

```
@n @verbatim
```

lookup3.c, by Bob Jenkins, May 2006, Public Domain.

These are functions for producing 32-bit hashes for hash table lookup. hashword(), hashlittle(), hashlittle2(), hashbig(), mix(), and final() are externally useful functions. Routines to test the hash are included if SELF_TEST is defined. You can use this free for any purpose. It's in the public domain. It has no warranty.

You probably want to use hashlittle(). hashlittle() and hashbig() hash byte arrays. hashlittle() is is faster than hashbig() on little-endian machines. Intel and AMD are little-endian machines. On second thought, you probably want hashlittle2(), which is identical to hashlittle() except it returns two 32-bit hashes for the price of one. You could implement hashbig2() if you wanted but I haven't bothered here.

If you want to find a hash of, say, exactly 7 integers, do a = i1; b = i2; c = i3; mix(a,b,c); a += i4; b += i5; c += i6; mix(a,b,c); a += i7; final(a,b,c); then use c as the hash value. If you have a variable length array of 4-byte integers to hash, use hashword(). If you have a byte array (like a character string), use hashlittle(). If you have several byte arrays, or a mix of things, see the comments above hashlittle().

Why is this so big? I read 12 bytes at a time into 3 4-byte integers, then mix those integers. This is fast (you can do a lot more thorough mixing with 12∗3 instructions on 3 integers than you can with 3 instructions

on 1 byte), but shoehorning those bytes into integers efficiently is messy.

Definition in file lookup3.c.

7.33 src/main.c File Reference

Application framework.

```
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <fcntl.h>
#include <sys/file.h>
#include "include/dtsapp.h"
#include "include/private.h"
```

Functions

- void printgnu ()

  Print a brief GNU copyright notice on console.
- void daemonize ()

  Daemonise the application using fork/exit.
- int lockpidfile (const char ∗runfile)

  Lock the run file in the framework application info.
- void framework_mkcore (char ∗progname, char ∗name, char ∗email, char ∗web, int year, char ∗runfile, int flags, syssighandler sigfunc)

  Initilise application data structure and return a reference.
- int framework_init (int argc, char ∗argv[], frameworkfunc callback)

  Initilise the application daemonise and join the manager thread.

7.33.1 Detailed Description

Application framework.

Definition in file main.c.

7.34 src/nf_ctrack.c File Reference

```
#include "config.h"
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/ioctl.h>
#include <netinet/in.h>
#include <linux/types.h>
#include <linux/netfilter.h>
#include <libnetfilter_conntrack/libnetfilter_conntrack.h>
#include <libnetfilter_conntrack/libnetfilter_conntrack_tcp.h>
#include "include/dtsapp.h"
#include "include/private.h"
```

Data Structures

- struct nfct_struct

Enumerations

- enum NF_CTRACK_FLAGS { NFCTRACK_DONE = 1 << 0 }

Functions

- uint8_t nf_ctrack_init (void)
- struct nf_conntrack * nf_ctrack_buildct (uint8_t *pkt)
- uint8_t nf_ctrack_delete (uint8_t *pkt)
- uint8_t nf_ctrack_nat (uint8_t *pkt, uint32_t addr, uint16_t port, uint8_t dnat)
- void nf_ctrack_dump (void)
- struct nfct_struct * nf_ctrack_trace (void)
- void nf_ctrack_endtrace (struct nfct_struct *nfct)
- void nf_ctrack_close (void)

Variables

- struct nfct_struct * ctrack = NULL

7.34.1 Enumeration Type Documentation

7.34.1.1 enum NF_CTRACK_FLAGS

Enumerator:

NFCTRACK_DONE

Definition at line 37 of file nf_ctrack.c.

```
                    {
    NFCTRACK_DONE     = 1 << 0
};
```

7.34.2 Function Documentation

7.34.2.1 struct nf_conntrack* nf_ctrack_buildct (uint8_t * *pkt*) [read]

Definition at line 90 of file nf_ctrack.c.

Referenced by nf_ctrack_delete(), and nf_ctrack_nat().

```
                                                                {
    struct nf_conntrack *ct;
    struct iphdr *ip = (struct iphdr *)pkt;
    union l4hdr *l4 = (union l4hdr *)(pkt + (ip->ihl * 4));

    if (!(ct = nfct_new())) {
        return (NULL);
    };

    /*Build tuple*/
    nfct_set_attr_u8(ct, ATTR_L3PROTO, PF_INET);
    nfct_set_attr_u32(ct, ATTR_IPV4_SRC, ip->saddr);
    nfct_set_attr_u32(ct, ATTR_IPV4_DST, ip->daddr);
    nfct_set_attr_u8(ct, ATTR_L4PROTO, ip->protocol);
    switch(ip->protocol) {
        case IPPROTO_TCP:
            nfct_set_attr_u16(ct, ATTR_PORT_SRC, l4->tcp.source);
            nfct_set_attr_u16(ct, ATTR_PORT_DST, l4->tcp.dest);
            break;
        case IPPROTO_UDP:
            nfct_set_attr_u16(ct, ATTR_PORT_SRC, l4->udp.source);
            nfct_set_attr_u16(ct, ATTR_PORT_DST, l4->udp.dest);
            break;
        case IPPROTO_ICMP:
            nfct_set_attr_u8(ct, ATTR_ICMP_TYPE, l4->icmp.type);
            nfct_set_attr_u8(ct, ATTR_ICMP_CODE, l4->icmp.code);
            nfct_set_attr_u16(ct, ATTR_ICMP_ID, l4->icmp.un.echo.id);
            /* no break */
        default
                :
            break;
    };

    return (ct);
}
```

7.34.2.2 void nf_ctrack_close (void)

Definition at line 275 of file nf_ctrack.c.

References ctrack, and objunref().

Referenced by nf_ctrack_delete(), nf_ctrack_dump(), and nf_ctrack_nat().

```
                                {
    if (ctrack) {
        objunref(ctrack);
    }
    ctrack = NULL;
}
```

7.34.2.3 uint8_t nf_ctrack_delete (uint8_t * *pkt*)

Definition at line 126 of file nf_ctrack.c.

References ctrack, nf_ctrack_buildct(), nf_ctrack_close(), nf_ctrack_init(), nfct_struct::nfct, objlock(), and objunlock().

```
                                                {
    struct nf_conntrack *ct;
    uint8_t unref = 0;
    uint8_t ret = 0;

    if (!ctrack) {
        if (nf_ctrack_init()) {
            return (-1);
        }
        unref = 1;
    }

    ct = nf_ctrack_buildct(pkt);
    objlock(ctrack);
    if (nfct_query(ctrack->nfct, NFCT_Q_DESTROY, ct) < 0) {
        ret = -1;
    }
    objunlock(ctrack);
    nfct_destroy(ct);

    if (unref) {
        nf_ctrack_close();
    }

    return (ret);
}
```

7.34.2.4 void nf_ctrack_dump (void)

Definition at line 197 of file nf_ctrack.c.

References ctrack, nf_ctrack_close(), nf_ctrack_init(), nfct_struct::nfct, objlock(), and objunlock().

```
                                    {
    uint32_t family = PF_INET;
    uint8_t unref = 0;

    if (!ctrack) {
        if (nf_ctrack_init()) {
            return;
        }
        unref = 1;
    }

    objlock(ctrack);
    nfct_callback_register(ctrack->nfct, NFCT_T_ALL, nfct_cb, NULL);
    nfct_query(ctrack->nfct, NFCT_Q_DUMP, &family);
    nfct_callback_unregister(ctrack->nfct);
    objunlock(ctrack);

    if (unref) {
        nf_ctrack_close();
    }
}
```

7.34.2.5 void nf_ctrack_endtrace (struct nfct_struct * *nfct*)

Definition at line 268 of file nf_ctrack.c.

References NFCTRACK_DONE, objunref(), and setflag.

```
                                                              {
    if (nfct) {
        setflag(nfct, NFCTRACK_DONE);
    }
    objunref(nfct);
}
```

7.34.2.6 uint8_t nf_ctrack_init (void)

Definition at line 83 of file nf_ctrack.c.

References ctrack.

Referenced by nf_ctrack_delete(), nf_ctrack_dump(), and nf_ctrack_nat().

```
                                    {
    if (!ctrack && !(ctrack = nf_ctrack_alloc(CONNTRACK, 0))) {
        return (-1);
    }
    return (0);
}
```

7.34.2.7 uint8_t nf_ctrack_nat (uint8_t * *pkt*, uint32_t *addr*, uint16_t *port*, uint8_t *dnat*)

Definition at line 153 of file nf_ctrack.c.

References ctrack, nf_ctrack_buildct(), nf_ctrack_close(), nf_ctrack_init(), nfct_struct::nfct, objlock(), and objunlock().

```
          {
    struct iphdr *ip = (struct iphdr *)pkt;
    struct nf_conntrack *ct;
    uint8_t unref = 0;
    uint8_t ret = 0;

    if (!ctrack) {
        if (nf_ctrack_init()) {
            return (-1);
        }
        unref = 1;
    }

    ct = nf_ctrack_buildct(pkt);
    nfct_setobjopt(ct, NFCT_SOPT_SETUP_REPLY);

    nfct_set_attr_u32(ct, ATTR_TIMEOUT, 120);
    nfct_set_attr_u32(ct, (dnat) ? ATTR_DNAT_IPV4 : ATTR_SNAT_IPV4, addr);

    switch(ip->protocol) {
        case IPPROTO_TCP:
            nfct_set_attr_u8(ct, ATTR_TCP_STATE, TCP_CONNTRACK_ESTABLISHED);
            /* no break */
        case IPPROTO_UDP:
            if (port) {
                nfct_set_attr_u16(ct, (dnat) ? ATTR_DNAT_PORT : ATTR_SNAT_PORT,
   port);
            }
            break;
    }

    objlock(ctrack);
    if (nfct_query(ctrack->nfct, NFCT_Q_CREATE_UPDATE, ct) < 0) {
        ret = -1;
    }
    objunlock(ctrack);
    nfct_destroy(ct);

    if (unref) {
        nf_ctrack_close();
    }

    return (ret);
}
```

7.34.2.8 struct nfct_struct* nf_ctrack_trace (void) [read]

Definition at line 254 of file nf_ctrack.c.

References framework_mkthread(), nfct_struct::nfct, and objunref().

```
                                        {
    struct nfct_struct *nfct;

    if (!(nfct = nf_ctrack_alloc(CONNTRACK, NFCT_ALL_CT_GROUPS))) {
        return (NULL);
    }

    if (!framework_mkthread(nf_ctrack_trace_th, NULL, NULL,
      nfct)) {
        objunref(nfct);
        return (NULL);
    }
    return (nfct);
}
```

7.34.3 Variable Documentation

7.34.3.1 struct nfct_struct * ctrack = NULL

Referenced by nf_ctrack_close(), nf_ctrack_delete(), nf_ctrack_dump(), nf_ctrack_init(), and nf_ctrack_nat().

7.35 src/nf_queue.c File Reference

```
#include "config.h"
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/ioctl.h>
#include <netinet/in.h>
#include <linux/types.h>
#include <linux/netfilter.h>
#include <libnetfilter_queue/libnetfilter_queue.h>
#include "include/dtsapp.h"
#include "include/private.h"
```

Data Structures

- struct nfq_struct
- struct nfq_queue
- struct nfq_list

Enumerations

- enum NF_QUEUE_FLAGS { NFQUEUE_DONE = 1 << 0 }

Functions

- struct nfq_queue * nfqueue_attach (uint16_t pf, uint16_t num, uint8_t mode, uint32_t range, nfqueue_cb cb, void *data)
- uint16_t snprintf_pkt (struct nfq_data *tb, struct nfqnl_msg_packet_hdr *ph, uint8_t *pkt, char *buff, uint16_t len)

Variables

- struct nfq_list * nfqueues = NULL

7.35.1 Enumeration Type Documentation

7.35.1.1 enum NF_QUEUE_FLAGS

Enumerator:

NFQUEUE_DONE

Definition at line 37 of file nf_queue.c.

```
                    {
    NFQUEUE_DONE     = 1 << 0
};
```

7.35.2 Function Documentation

7.35.2.1 struct nfq_queue* nfqueue_attach (uint16_t *pf*, uint16_t *num*, uint8_t *mode*, uint32_t *range*, nfqueue_cb *cb*, void * *data*) [read]

Definition at line 225 of file nf_queue.c.

References bucket_list_find_key(), nfq_queue::cb, nfq_queue::data, nfq_struct::h, nfq_queue::nfq, nfqueues, objalloc(), objlock(), objunlock(), objunref(), nfq_queue::qh, and nfq_list::queues.

```
                                                    {
    struct nfq_queue *nfq_q;

    if (!(nfq_q = objalloc(sizeof(*nfq_q), nfqueue_close_q))) {
        return (NULL);
    }

    objlock(nfqueues);
    if (!(nfqueues && (nfq_q->nfq = bucket_list_find_key
      (nfqueues->queues, &pf))) &&
            !(nfq_q->nfq || (nfq_q->nfq = nfqueue_init(pf)))) {
        objunlock(nfqueues);
        objunref(nfq_q);
        return (NULL);
    }
    objunlock(nfqueues);

    if (!(nfq_q->qh = nfq_create_queue(nfq_q->nfq->h, num, &
      nfqueue_callback, nfq_q))) {
        objunref(nfq_q);
        return (NULL);
    }

    if (cb) {
        nfq_q->cb = cb;
    }

    if (data) {
        nfq_q->data = data;
    }

    nfq_set_mode(nfq_q->qh, mode, range);

    return (nfq_q);
}
```

7.35.2.2 uint16_t snprintf_pkt (struct nfq_data * *tb*, struct nfqnl_msg_packet_hdr * *ph*, uint8_t * *pkt*, char * *buff*, uint16_t *len*)

Definition at line 259 of file nf_queue.c.

References id.

```
                                                {
    struct iphdr *ip = (struct iphdr *)pkt;
    char *tmp = buff;
    uint32_t id, mark, ifi;
    uint16_t tlen, left = len;
    char saddr[INET_ADDRSTRLEN], daddr[INET_ADDRSTRLEN];

    if (ph) {
        id = ntohl(ph->packet_id);
        snprintf(tmp, left, "hw_protocol=0x%04x hook=%u id=%u ",
                 ntohs(ph->hw_protocol), ph->hook, id);
        tlen = strlen(tmp);
        tmp += tlen;
        left -= tlen;
    }

    if ((mark = nfq_get_nfmark(tb))) {
        snprintf(tmp, left, "mark=%u ", mark);
        tlen = strlen(tmp);
        tmp += tlen;
        left -= tlen;
    }

    if ((ifi = nfq_get_indev(tb))) {
        snprintf(tmp, left, "indev=%u ", ifi);
        tlen = strlen(tmp);
        tmp += tlen;
        left -= tlen;
    }

    if ((ifi = nfq_get_outdev(tb))) {
        snprintf(tmp, left, "outdev=%u ", ifi);
        tlen = strlen(tmp);
        tmp += tlen;
        left -= tlen;
    }

    if (pkt && (ip->version == 4)) {
        union l4hdr *l4 = (union l4hdr *)(pkt + (ip->ihl*4));

        inet_ntop(AF_INET, &ip->saddr, saddr, INET_ADDRSTRLEN);
        inet_ntop(AF_INET, &ip->daddr, daddr, INET_ADDRSTRLEN);

        snprintf(tmp, left, "src=%s dst=%s proto=%i ", saddr, daddr, ip->
      protocol);
        tlen = strlen(tmp);
        tmp += tlen;
        left -= tlen;

        switch(ip->protocol) {
            case IPPROTO_TCP:
                snprintf(tmp, left, "sport=%i dport=%i ", ntohs(l4->tcp.source)
      , ntohs(l4->tcp.dest));
                break;
            case IPPROTO_UDP:
                snprintf(tmp, left, "sport=%i dport=%i ", ntohs(l4->udp.source)
      , ntohs(l4->udp.dest));
                break;
            case IPPROTO_ICMP:
                snprintf(tmp, left, "type=%i code=%i id=%i ", l4->icmp.type, l4
    ->icmp.code, ntohs(l4->icmp.un.echo.id));
                break;
        }
        tlen = strlen(tmp);
        tmp += tlen;
        left -= tlen;
    }

    return (len - left);
}
```

7.35.3 Variable Documentation

7.35.3.1 struct nfq_list * nfqueues = NULL

Referenced by nfqueue_attach().

7.36 src/openldap.c File Reference

```
#include <ldap.h>
#include <ldap_schema.h>
#include <lber.h>
#include <sasl/sasl.h>
#include <stdlib.h>
#include <stdint.h>
#include <stdio.h>
#include <ctype.h>
#include <sys/time.h>
#include <stdarg.h>
#include "include/dtsapp.h"
```

Data Structures

- struct sasl_defaults
- struct ldap_simple
- struct ldap_conn
- struct ldap_modify
- struct ldap_add
- struct ldap_modval
- struct ldap_modreq

Functions

- int ldap_count (LDAP *ld, LDAPMessage *message, int *err)
- struct ldap_entry * ldap_getent (LDAP *ld, LDAPMessage **msgptr, LDAPMessage *result, int b64enc, int *err)
- int dts_sasl_interact (LDAP *ld, unsigned flags, void *defaults, void *in)
- void free_simple (void *data)
- void free_modval (void *data)
- void free_modreq (void *data)
- void free_modify (void *data)
- void free_add (void *data)
- void free_sasl (void *data)
- void free_ldapconn (void *data)
- void free_result (void *data)
- void free_entry (void *data)
- void free_rdnarr (void *data)
- void free_rdn (void *data)
- void free_attr (void *data)
- void free_attrvalarr (void *data)
- void free_attrval (void *data)
- void free_entarr (void *data)
- int modify_hash (const void *data, int key)
- int ldap_rebind_proc (LDAP *ld, LDAP_CONST char *url, ber_tag_t request, ber_int_t msgid, void *params)
- struct ldap_conn * ldap_connect (const char *uri, enum ldap_starttls starttls, int timelimit, int limit, int debug, int *err)
- int ldap_simplebind (struct ldap_conn *ld, const char *dn, const char *passwd)
- int ldap_simplerebind (struct ldap_conn *ldap, const char *initialdn, const char *initialpw, const char *base, const char *filter, const char *uidrdn, const char *uid, const char *passwd)

- int ldap_saslbind (struct ldap_conn *ld, const char *mech, const char *realm, const char *authcid, const char *passwd, const char *authzid)
- void ldap_close (struct ldap_conn *ld)
- const char * ldap_errmsg (int res)
- int searchresults_hash (const void *data, int key)
- struct ldap_results * dts_ldapsearch (struct ldap_conn *ld, const char *base, int scope, const char *filter, char *attrs[], int b64enc, int *err)
- struct ldap_results * ldap_search_sub (struct ldap_conn *ld, const char *base, const char *filter, int b64enc, int *res,...)
- struct ldap_results * ldap_search_one (struct ldap_conn *ld, const char *base, const char *filter, int b64enc, int *res,...)
- struct ldap_results * ldap_search_base (struct ldap_conn *ld, const char *base, const char *filter, int b64enc, int *res,...)
- char * ldap_getdn (LDAP *ld, LDAPMessage *message, int *err)
- char * ldap_getattribute (LDAP *ld, LDAPMessage *message, BerElement **berptr, int *err)
- char * ldap_encattr (void *attrval, int b64enc, enum ldap_attrtype *type)
- struct berval ** ldap_attrvals (LDAP *ld, LDAPMessage *message, char *attr, int *cnt, int *err)
- int ldapattr_hash (const void *data, int key)
- struct bucket_list * attr2bl (LDAP *ld, LDAPMessage *message, struct ldap_attr **first, int b64enc, int *res)
- void ldap_unref_attr (struct ldap_entry *entry, struct ldap_attr *attr)
- void ldap_unref_entry (struct ldap_results *results, struct ldap_entry *entry)
- struct ldap_entry * ldap_getentry (struct ldap_results *results, const char *dn)
- struct ldap_attr * ldap_getattr (struct ldap_entry *entry, const char *attr)
- struct ldap_modify * ldap_modifyinit (const char *dn)
- struct ldap_modreq * new_modreq (struct bucket_list *modtype, const char *attr)
- struct ldap_modreq * getmodreq (struct ldap_modify *lmod, const char *attr, int modop)
- int add_modifyval (struct ldap_modreq *modr, const char *value)
- int ldap_mod_del (struct ldap_modify *lmod, const char *attr,...)
- int ldap_mod_add (struct ldap_modify *lmod, const char *attr,...)
- int ldap_mod_rep (struct ldap_modify *lmod, const char *attr,...)
- LDAPMod * ldap_reqtoarr (struct ldap_modreq *modr, int type)
- int ldap_domodify (struct ldap_conn *ld, struct ldap_modify *lmod)
- int ldap_mod_delattr (struct ldap_conn *ldap, const char *dn, const char *attr, const char *value)
- int ldap_mod_remattr (struct ldap_conn *ldap, const char *dn, const char *attr)
- int ldap_mod_addattr (struct ldap_conn *ldap, const char *dn, const char *attr, const char *value)
- int ldap_mod_repattr (struct ldap_conn *ldap, const char *dn, const char *attr, const char *value)
- struct ldap_add * ldap_addinit (const char *dn)
- struct ldap_modreq * getaddreq (struct ldap_add *ladd, const char *attr)
- int ldap_add_attr (struct ldap_add *ladd, const char *attr,...)
- int ldap_doadd (struct ldap_conn *ld, struct ldap_add *ladd)

7.36.1 Function Documentation

7.36.1.1 int add_modifyval (struct ldap_modreq * *modr*, const char * *value*)

Definition at line 1160 of file openldap.c.

References ALLOC_CONST, ldap_modreq::cnt, ldap_modreq::first, free_modval(), ldap_modreq::last, ldap_modval::next, objalloc(), objunref(), and ldap_modval::value.

Referenced by ldap_add_attr(), ldap_mod_add(), ldap_mod_del(), and ldap_mod_rep().

```
                                                                {
    struct ldap_modval *newval;

    if (!(newval = objalloc(sizeof(*newval), free_modval)))
      {
        return 1;
    }

    ALLOC_CONST(newval->value, value);
    if (!newval->value) {
        objunref(newval);
        return 1;
    }

    if (!modr->first) {
        modr->first = newval;
    }
    if (modr->last) {
        modr->last->next = newval;
    }
    modr->cnt++;
    modr->last = newval;

    return 0;
}
```

7.36.1.2 struct bucket_list* attr2bl (LDAP * *ld,* LDAPMessage * *message,* struct ldap_attr ** *first,* int *b64enc,* int * *res*) [read]

Definition at line 827 of file openldap.c.

References addtobucket(), ldap_attrval::buffer, ldap_attr::count, create_bucketlist(), free_attr(), free_attrval(), free_attrvalarr(), ldap_attrvals(), ldap_encattr(), ldap_getattribute(), ldapattr_hash(), ldap_attrval::len, ldap_attr::name, ldap_attr::next, objalloc(), objunref(), ldap_attr::prev, ldap_attrval::type, and ldap_attr::vals.

Referenced by ldap_getent().

```
                                    {
    BerElement *ber = NULL;
    struct bucket_list *bl;
    struct ldap_attr *la, *prev = NULL;
    struct ldap_attrval *lav, **lavals;
    struct berval **tmp, **vals = NULL;
    enum ldap_attrtype type;
    char *attr;
    int cnt;
    char *eval;

    if (!(bl = create_bucketlist(4, ldapattr_hash
      ))) {
        if (res) {
            *res = LDAP_NO_MEMORY;
        }
        return NULL;
    }

    while((attr = ldap_getattribute(ld, message, &ber, res)))
      {
        tmp = vals = ldap_attrvals(ld, message, attr, &cnt, res);
        la = objalloc(sizeof(*la), free_attr);
        if (first && !*first) {
            *first = la;
        }
        la->next = NULL;
        if (prev) {
            prev->next = la;
            la->prev = prev;
        } else {
            la->prev = NULL;
        }
        prev = la;
        lavals = objalloc(sizeof(void *) * (cnt+1), free_attrvalarr
      );
        if (!lavals || !la) {
            if (res) {
                *res = LDAP_NO_MEMORY;
            }
            if (la) {
                objunref(la);
```

```
            }
            if (lavals) {
                objunref(lavals);
            }
            objunref(bl);
            ldap_value_free_len(vals);
            if (ber) {
                ber_free(ber, 0);
            }
            return NULL;
        }
        la->vals = lavals;
        la->name = attr;
        la->count = cnt;

        for(; *tmp; tmp++) {
            struct berval *bval = *tmp;

            *lavals = lav = objalloc(sizeof(*lav), free_attrval
    );
            lavals++;

            eval = ldap_encattr(bval, b64enc, &type);
            if (!eval || !lav) {
                if (res) {
                    *res = LDAP_NO_MEMORY;
                }
                objunref(bl);
                objunref(la);
                if (eval) {
                    objunref(eval);
                }
                ldap_value_free_len(vals);
                if (ber) {
                    ber_free(ber, 0);
                }
                return NULL;
            }
            lav->len = bval->bv_len;
            lav->buffer = eval;
            lav->type = type;
        }
        *lavals = NULL;
        ldap_value_free_len(vals);
        addtobucket(bl, la);
        objunref(la);
    }
    if (ber) {
        ber_free(ber, 0);
    }
    return bl;
}
```

7.36.1.3 struct ldap_results* dts_ldapsearch (struct ldap_conn * *ld*, const char * *base*, int *scope*, const char * *filter*, char * *attrs[]*, int *b64enc*, int * *err*) [read]

Definition at line 529 of file openldap.c.

References addtobucket(), ldap_results::count, create_bucketlist(), ldap_results::entries, ldap_results::first_entry, free_result(), ldap_conn::ldap, ldap_count(), ldap_getent(), ldap_conn::limit, ldap_entry::next, objalloc(), objlock(), objref(), objunlock(), objunref(), ldap_entry::prev, ldap_conn::sctrlsp, searchresults_hash(), and ldap_conn::timelim.

Referenced by ldap_search_base(), ldap_search_one(), and ldap_search_sub().

```
{
    struct timeval timeout = {0,0};
    struct ldap_results *results;
    struct ldap_entry *lent, *prev = NULL;
    LDAPMessage *result, *message = NULL;
    int res = LDAP_SUCCESS;

    if (!objref(ld)) {
        if (err) {
            *err = LDAP_UNAVAILABLE;
        }
        if (attrs) {
            free(attrs);
        }
        return NULL;
```

```
    }

    if ((results = objalloc(sizeof(*results), free_result)))
       {
        results->entries = create_bucketlist(4,
      searchresults_hash);
    }

    timeout.tv_sec = ld->timelim;
    timeout.tv_usec = 0;

    objlock(ld);
    if (!results || !results->entries ||
            (res = ldap_search_ext_s(ld->ldap, base, scope, filter, attrs
      , 0, ld->sctrlsp, NULL, &timeout, ld->limit, &result))) {
        objunlock(ld);
        objunref(ld);
        objunref(results);
        ldap_msgfree(result);
        if (err) {
            *err = (!results || !results->entries) ? LDAP_NO_MEMORY :
      res;
        }
        if (attrs) {
            free(attrs);
        }
        return NULL;
    }
    objunlock(ld);

    if (attrs) {
        free(attrs);
    }

    if ((results->count = ldap_count(ld->ldap, result, err))
       < 0) {
        objunref(ld);
        objunref(results);
        ldap_msgfree(result);
        return NULL;
    }

    while((lent = ldap_getent(ld->ldap, &message, result, b64enc
      , err))) {
        if (!results->first_entry) {
            results->first_entry = lent;
        }
        if (!addtobucket(results->entries, lent)) {
            res = LDAP_NO_MEMORY;
            objunref(lent);
            break;
        }
        lent->next = NULL;
        if (prev) {
            prev->next = lent;
            lent->prev = prev;
        } else {
            lent->prev = NULL;
        }
        prev = lent;
        objunref(lent);
    }
    ldap_msgfree(result);

    if (err) {
        *err = res;
    }

    if (res) {
        objunref(results);
        results = NULL;
    }

    objunref(ld);
    return results;
}
```

7.36.1.4 int dts_sasl_interact (LDAP * *ld*, unsigned *flags*, void * *defaults*, void * *in*)

Definition at line 372 of file openldap.c.

Referenced by ldap_rebind_proc(), and ldap_saslbind().

```
                                                                    {
    sasl_interact_t *interact = in;

    if (!ld) {
        return LDAP_PARAM_ERROR;
    }

    while( interact->id != SASL_CB_LIST_END ) {
        int rc = interaction(flags, interact, defaults);
        if (rc)  {
            return rc;
        }
        interact++;
    }
    return LDAP_SUCCESS;
}
```

7.36.1.5 void free_add (void * *data*)

Definition at line 116 of file openldap.c.

References ldap_add::bl, ldap_add::dn, and objunref().

Referenced by ldap_addinit().

```
                          {
    struct ldap_add *lmod = data;

    if (lmod->dn) {
        free((void *)lmod->dn);
    }

    if (lmod->bl) {
        objunref(lmod->bl);
    }
}
```

7.36.1.6 void free_attr (void * *data*)

Definition at line 222 of file openldap.c.

References ldap_attr::name, ldap_attr::next, objunref(), ldap_attr::prev, and ldap_attr::vals.

Referenced by attr2bl().

```
                           {
    struct ldap_attr *la = data;
    if (la->next) {
        la->next->prev = la->prev;
    }
    if (la->prev) {
        la->prev->next = la->next;
    }
    ldap_memfree((char *)la->name);
    if (la->vals) {
        objunref(la->vals);
    }
}
```

7.36.1.7 void free_attrval (void * *data*)

Definition at line 243 of file openldap.c.

References ldap_attrval::buffer, and objunref().

Referenced by attr2bl().

```
                              {
    struct ldap_attrval *av = data;
    if (av->buffer) {
        objunref(av->buffer);
    }
}
```

7.36.1.8 void free_attrvalarr (void * *data*)

Definition at line 236 of file openldap.c.

References objunref().

Referenced by attr2bl().

```
                                    {
    struct ldap_attrval **av = data;
    for(; *av; av++) {
        objunref(*av);
    }
}
```

7.36.1.9 void free_entarr (void * *data*)

Definition at line 250 of file openldap.c.

References objunref().

```
                                {
    struct ldap_entry **entarr = data;

    for(; *entarr; entarr++) {
        objunref(*entarr);
    }
}
```

7.36.1.10 void free_entry (void * *data*)

Definition at line 173 of file openldap.c.

References ldap_entry::attrs, ldap_entry::dn, ldap_entry::dnufn, ldap_entry::first_attr, ldap_attr::next, ldap_entry::next, objunref(), ldap_entry::prev, and ldap_entry::rdn.

Referenced by ldap_getent().

```
                               {
    struct ldap_entry *ent = data;
    struct ldap_attr *la;

    if (ent->prev) {
        ent->prev->next = ent->next;
    }
    if (ent->next) {
        ent->next->prev = ent->prev;
    }

    if (ent->dn) {
        ldap_memfree((void *)ent->dn);
    }
    if (ent->rdn) {
        objunref(ent->rdn);
    }
    if (ent->dnufn) {
        free((void *)ent->dnufn);
    }
    if (ent->attrs) {
        objunref(ent->attrs);
    }
    if (ent->first_attr) {
        for(la = ent->first_attr; la; la = la->next) {
            objunref(la);
        }
    }
}
```

7.36.1.11 void free_ldapconn (void * *data*)

Definition at line 148 of file openldap.c.

References ldap_conn::ldap, objunref(), ldap_conn::sasl, ldap_conn::sctrlsp, ldap_conn::simple, and ldap_conn::uri.

Referenced by ldap_connect().

```
                                {
    struct ldap_conn *ld = data;


    if (ld->uri) {
        free(ld->uri);
    }
    if (ld->ldap) {
        ldap_unbind_ext_s(ld->ldap, ld->sctrlsp, NULL);
    }
    if (ld->sasl) {
        objunref(ld->sasl);
    }
    if (ld->simple) {
        objunref(ld->simple);
    }
}
```

7.36.1.12 void free_modify (void * *data*)

Definition at line 102 of file openldap.c.

References ldap_modify::bl, ldap_modify::dn, and objunref().

Referenced by ldap_modifyinit().

```
                                {
    struct ldap_modify *lmod = data;
    int cnt;
    if (lmod->dn) {
        free((void *)lmod->dn);
    }

    for(cnt=0; cnt < 3; cnt++) {
        if (lmod->bl[cnt]) {
            objunref(lmod->bl[cnt]);
        }
    }
}
```

7.36.1.13 void free_modreq (void * *data*)

Definition at line 90 of file openldap.c.

References ldap_modreq::attr, ldap_modreq::first, ldap_modval::next, and objunref().

Referenced by new_modreq().

```
                                {
    struct ldap_modreq *modr = data;
    struct ldap_modval *modv;

    if (modr->attr) {
        free((void *)modr->attr);
    }
    for(modv = modr->first; modv; modv = modv->next) {
        objunref(modv);
    }
}
```

7.36.1.14 void free_modval (void * *data*)

Definition at line 82 of file openldap.c.

References ldap_modval::value.

Referenced by add_modifyval().

```
                                {
    struct ldap_modval *modv = data;

    if (modv->value) {
        free((void *)modv->value);
    }
}
```

7.36.1.15 void free_rdn (void * *data*)

Definition at line 211 of file openldap.c.

References ldap_rdn::name, objunref(), and ldap_rdn::value.

Referenced by ldap_getent().

```
                           {
    struct ldap_rdn *rdn = data;

    if (rdn->name) {
        objunref((void *)rdn->name);
    }
    if (rdn->value) {
        objunref((void *)rdn->value);
    }
}
```

7.36.1.16 void free_rdnarr (void * *data*)

Definition at line 203 of file openldap.c.

References objunref().

Referenced by ldap_getent().

```
                                {
    struct ldap_rdn **rdn = data;

    for(; *rdn; rdn++) {
        objunref(*rdn);
    }
}
```

7.36.1.17 void free_result (void * *data*)

Definition at line 166 of file openldap.c.

References ldap_results::entries, and objunref().

Referenced by dts_ldapsearch().

```
                                {
    struct ldap_results *res = data;
    if (res->entries) {
        objunref(res->entries);
    }
}
```

7.36.1.18 void free_sasl (void * *data*)

Definition at line 128 of file openldap.c.

References sasl_defaults::authcid, sasl_defaults::authzid, sasl_defaults::mech, sasl_defaults::passwd, and sasl_-defaults::realm.

Referenced by ldap_saslbind().

```
                          {
    struct sasl_defaults *sasl = data;

    if (sasl->mech) {
        free((void *)sasl->mech);
    }
    if (sasl->realm) {
        free((void *)sasl->realm);
    }
    if (sasl->authcid) {
        free((void *)sasl->authcid);
    }
    if (sasl->passwd) {
        free((void *)sasl->passwd);
    }
    if (sasl->authzid) {
        free((void *)sasl->authzid);
    }
}
```

7.36.1.19 void free_simple (void * *data*)

Definition at line 67 of file openldap.c.

References ldap_simple::cred, and ldap_simple::dn.

Referenced by ldap_simplebind().

```
                           {
    struct ldap_simple *simple = data;
    struct berval *bv = simple->cred;

    if (bv && bv->bv_val) {
        free(bv->bv_val);
    }
    if (bv) {
        free(bv);
    }
    if (simple->dn) {
        free((void *)simple->dn);
    }
}
```

7.36.1.20 struct ldap_modreq* getaddreq (struct ldap_add * *ladd,* const char * *attr*) [read]

Definition at line 1416 of file openldap.c.

References ldap_add::bl, bucket_list_find_key(), and new_modreq().

Referenced by ldap_add_attr().

```
                                                                    {
    struct bucket_list *bl = ladd->bl;
    struct ldap_modreq *modr = NULL;

    if (bl && !(modr = bucket_list_find_key(bl, attr)))
       {
        if (!(modr = new_modreq(bl, attr))) {
            return NULL;
        }
    }
    return modr;
}
```

7.36.1.21 struct ldap_modreq* getmodreq (struct ldap_modify * *lmod,* const char * *attr,* int *modop*) [read]

Definition at line 1136 of file openldap.c.

References ldap_modify::bl, bucket_list_find_key(), and new_modreq().

Referenced by ldap_mod_add(), ldap_mod_del(), and ldap_mod_rep().

```
        {
    struct bucket_list *bl = NULL;
    struct ldap_modreq *modr = NULL;

    switch (modop) {
        case LDAP_MOD_REPLACE:
            bl = lmod->bl[0];
            break;
        case LDAP_MOD_DELETE:
            bl = lmod->bl[1];
            break;
        case LDAP_MOD_ADD:
            bl = lmod->bl[2];
            break;
    }

    if (bl && !(modr = bucket_list_find_key(bl, attr)))
       {
        if (!(modr = new_modreq(bl, attr))) {
            return NULL;
        }
    }
    return modr;
}
```

7.36.1.22 int ldap_add_attr (struct ldap_add * *ladd,* const char * *attr,* ...)

Definition at line 1428 of file openldap.c.

References add_modifyval(), getaddreq(), and objunref().

```
                                                                    {
    va_list a_list;
    char *val;
    struct ldap_modreq *modr;

    if (!(modr = getaddreq(ladd, attr))) {
        return 1;
    }

    va_start(a_list, attr);
    while((val = va_arg(a_list, void *))) {
        if (add_modifyval(modr, val)) {
            objunref(modr);
            return(1);
        }
    }

    objunref(modr);
    va_end(a_list);
    return 0;
}
```

7.36.1.23 struct ldap_add* ldap_addinit (const char * *dn*) [read]

Definition at line 1395 of file openldap.c.

References ALLOC_CONST, ldap_add::bl, create_bucketlist(), ldap_add::dn, free_add(), modify_hash(), objalloc(), and objunref().

```
                                                {
    struct ldap_add *mod;

    if (!(mod = objalloc(sizeof(*mod), free_add))) {
```

```
        return NULL;
    }

    ALLOC_CONST(mod->dn, dn);
    if (!mod->dn) {
        objunref(mod);
        return NULL;
    }

    if (!(mod->bl = create_bucketlist(4, modify_hash
      ))) {
        objunref(mod);
        return NULL;
    }

    return mod;
}
```

7.36.1.24 struct berval** ldap_attrvals (LDAP * *ld*, LDAPMessage * *message*, char * *attr*, int * *cnt*, int * *err*) [read]

Definition at line 790 of file openldap.c.

References objlock(), and objunlock().

Referenced by attr2bl().

```
                    {
    struct berval **vals = NULL;

    objlock(ld);
    vals = ldap_get_values_len(ld, message, attr);
    objunlock(ld);

    if (cnt) {
        *cnt = ldap_count_values_len(vals);
    }

    if (!err) {
        return vals;
    }

    if (!vals) {
        ldap_get_option(ld, LDAP_OPT_RESULT_CODE, err);
    } else {
        *err = LDAP_SUCCESS;
    }

    return vals;
}
```

7.36.1.25 void ldap_close (struct ldap_conn * *ld*)

Definition at line 508 of file openldap.c.

References objunref().

```
                                            {
    objunref(ld);
}
```

7.36.1.26 struct ldap_conn* ldap_connect (const char * *uri*, enum ldap_starttls *starttls*, int *timelimit*, int *limit*, int *debug*, int * *err*) [read]

Definition at line 295 of file openldap.c.

References free_ldapconn(), ldap_conn::ldap, ldap_rebind_proc(), LDAP_STARTTLS_ENFORCE, LDAP_STARTTLS_NONE, ldap_conn::limit, objalloc(), objunref(), ldap_conn::sasl, ldap_conn::sctrlsp, ldap_conn::timelim, and ldap_conn::uri.

```
                                                                {
    struct ldap_conn *ld;
    int version = 3;
    int res, sslres;
    struct timeval timeout;

    if (!(ld = objalloc(sizeof(*ld), free_ldapconn))) {
        return NULL;
    }

    ld->uri = strdup(uri);
    ld->sctrlsp = NULL;
    ld->timelim = timelimit;
    ld->limit = limit;
    ld->sasl = NULL;

    if ((res = ldap_initialize(&ld->ldap, ld->uri) != LDAP_SUCCESS)) {
        objunref(ld);
        ld = NULL;
    } else {
        if (debug) {
            ldap_set_option(NULL, LDAP_OPT_DEBUG_LEVEL, &debug);
            ber_set_option(NULL, LBER_OPT_DEBUG_LEVEL, &debug);
        }
        if (timelimit) {
            timeout.tv_sec = timelimit;
            timeout.tv_usec = 0;
            ldap_set_option(ld->ldap, LDAP_OPT_NETWORK_TIMEOUT, (void *)&
      timeout);
        }
        ldap_set_option(ld->ldap, LDAP_OPT_PROTOCOL_VERSION, &version);
        ldap_set_option(ld->ldap, LDAP_OPT_REFERRALS, (void *)LDAP_OPT_ON);
        ldap_set_rebind_proc(ld->ldap, ldap_rebind_proc, ld
      );

        if ((starttls != LDAP_STARTTLS_NONE) & !
      ldap_tls_inplace(ld->ldap) && (sslres = ldap_start_tls_s(ld->ldap, ld->sctrlsp
      , NULL))) {
            if (starttls == LDAP_STARTTLS_ENFORCE) {
                objunref(ld);
                ld = NULL;
                res = sslres;
            }
        }
    }
    *err = res;
    return ld;
}
```

7.36.1.27 int ldap_count (LDAP * *ld*, LDAPMessage * *message*, int * *err*)

Definition at line 693 of file openldap.c.

References objlock(), and objunlock().

Referenced by dts_ldapsearch().

```
                                                                {
    int x;

    objlock(ld);
    x = ldap_count_entries(ld, message);
    objunlock(ld);

    if (!err) {
        return x;
    }

    if (x < 0) {
        objlock(ld);
        ldap_get_option(ld, LDAP_OPT_RESULT_CODE, err);
        objunlock(ld);
    } else {
        *err = LDAP_SUCCESS;
    }
    return x;
}
```

7.36.1.28 int ldap_doadd (struct ldap_conn * *ld*, struct ldap_add * *ladd*)

Definition at line 1450 of file openldap.c.

References ldap_add::bl, bucket_list_cnt(), ldap_add::dn, init_bucket_loop(), ldap_conn::ldap, ldap_reqtoarr(), next_bucket_loop(), objlock(), objunlock(), objunref(), ldap_conn::sctrlsp, and stop_bucket_loop().

```
                                                                    {
    struct bucket_loop *bloop;
    struct ldap_modreq *modr;
    LDAPMod **modarr, **tmp, *item;
    int tot=0, res;

    tot = bucket_list_cnt(ladd->bl);
    tmp = modarr = calloc(sizeof(void *), (tot+1));

    bloop = init_bucket_loop(ladd->bl);
    while(bloop && ((modr = next_bucket_loop(bloop)))) {
        if (!(item = ldap_reqtoarr(modr, -1))) {
            ldap_mods_free(modarr, 1);
            return LDAP_NO_MEMORY;
        }
        *tmp = item;
        tmp++;
        objunref(modr);
    }
    stop_bucket_loop(bloop);
    *tmp = NULL;

    objlock(ld);
    res = ldap_modify_ext_s(ld->ldap, ladd->dn, modarr, ld->sctrlsp
      , NULL);
    objunlock(ld);
    ldap_mods_free(modarr, 1);

    return res;
}
```

7.36.1.29 int ldap_domodify (struct ldap_conn * *ld*, struct ldap_modify * *lmod*)

Definition at line 1299 of file openldap.c.

References ldap_modify::bl, bucket_list_cnt(), ldap_modreq::cnt, ldap_modify::dn, init_bucket_loop(), ldap_conn::ldap, ldap_reqtoarr(), next_bucket_loop(), objlock(), objref(), objunlock(), objunref(), ldap_conn::sctrlsp, and stop_bucket_loop().

Referenced by ldap_mod_addattr(), ldap_mod_delattr(), and ldap_mod_repattr().

```
                                                                {
    struct bucket_loop *bloop;
    struct ldap_modreq *modr;
    LDAPMod **modarr, **tmp, *item;
    int cnt, tot=0, res;

    if (!objref(ld)) {
        return LDAP_UNAVAILABLE;
    }

    for(cnt = 0; cnt < 3; cnt++) {
        tot += bucket_list_cnt(lmod->bl[cnt]);
    }
    tmp = modarr = calloc(sizeof(void *), (tot+1));

    for(cnt = 0; cnt < 3; cnt++) {
        bloop = init_bucket_loop(lmod->bl[cnt]);
        while(bloop && ((modr = next_bucket_loop(bloop)))) {
            if (!(item = ldap_reqtoarr(modr, cnt))) {
                ldap_mods_free(modarr, 1);
                objunref(ld);
                return LDAP_NO_MEMORY;
            }
            *tmp = item;
            tmp++;
            objunref(modr);
        }
        stop_bucket_loop(bloop);
    }
    *tmp = NULL;
```

```
    objlock(ld);
    res = ldap_modify_ext_s(ld->ldap, lmod->dn, modarr, ld->sctrlsp
      , NULL);
    objunlock(ld);
    ldap_mods_free(modarr, 1);
    objunref(ld);
    return res;
}
```

7.36.1.30 char* ldap_encattr (void * *attrval,* int *b64enc,* enum ldap_attrtype * *type*)

Definition at line 761 of file openldap.c.

References b64enc_buf(), LDAP_ATTRTYPE_B64, LDAP_ATTRTYPE_CHAR, LDAP_ATTRTYPE_OCTET, objalloc(), and objsize().

Referenced by attr2bl().

```
                                                                    {
    struct berval *val = attrval;
    char *aval = NULL;
    int len, pos, atype;

    len = val->bv_len;
    for(pos=0; isprint(val->bv_val[pos]); pos++)
        ;
    if (pos == len) {
        aval = objalloc(val->bv_len+1, NULL);
        strncpy(aval, val->bv_val, objsize(aval));
        atype = LDAP_ATTRTYPE_CHAR;
    } else
        if (b64enc) {
            aval = b64enc_buf(val->bv_val, val->bv_len, 0);
            atype = LDAP_ATTRTYPE_B64;
        } else {
            aval = objalloc(val->bv_len, NULL);
            memcpy(aval, val->bv_val, objsize(aval));
            atype = LDAP_ATTRTYPE_OCTET;
        }

    if (type) {
        *type = atype;
    }

    return aval;
}
```

7.36.1.31 const char* ldap_errmsg (int *res*)

Definition at line 512 of file openldap.c.

```
                                        {
    return ldap_err2string(res);
}
```

7.36.1.32 struct ldap_attr* ldap_getattr (struct ldap_entry * *entry,* const char * *attr*) [read]

Definition at line 1090 of file openldap.c.

References ldap_entry::attrs, and bucket_list_find_key().

```
        {
    if (!entry || !entry->attrs) {
        return NULL;
    }
    return (struct ldap_attr *)bucket_list_find_key
      (entry->attrs, attr);
}
```

7.36.1.33 char* ldap_getattribute (LDAP * *ld*, LDAPMessage * *message*, BerElement ** *berptr*, int * *err*)

Definition at line 736 of file openldap.c.

References objlock(), and objunlock().

Referenced by attr2bl().

```
          {
    BerElement *ber = *berptr;
    char *attr = NULL;

    objlock(ld);
    if (ber) {
        attr = ldap_next_attribute(ld, message, ber);
    } else {
        attr = ldap_first_attribute(ld, message, berptr);
    }
    if (!err) {
        objunlock(ld);
        return attr;
    }

    if (!attr) {
        ldap_get_option(ld, LDAP_OPT_RESULT_CODE, err);
    } else {
        *err = LDAP_SUCCESS;
    }

    objunlock(ld);
    return attr;
}
```

7.36.1.34 char* ldap_getdn (LDAP * *ld*, LDAPMessage * *message*, int * *err*)

Definition at line 714 of file openldap.c.

References objlock(), and objunlock().

Referenced by ldap_getent().

```
                                                    {
    char *dn;

    objlock(ld);
    dn = ldap_get_dn(ld, message);
    objunlock(ld);

    if (!err) {
        return dn;
    }

    if (!dn) {
        objlock(ld);
        ldap_get_option(ld, LDAP_OPT_RESULT_CODE, err);
        objunlock(ld);
    } else {
        *err = LDAP_SUCCESS;
    }

    return dn;
}
```

7.36.1.35 struct ldap_entry * ldap_getent (LDAP * *ld*, LDAPMessage ** *msgptr*, LDAPMessage * *result*, int *b64enc*, int * *err*) [read]

Definition at line 918 of file openldap.c.

References ALLOC_CONST, attr2bl(), ldap_entry::attrs, ldap_entry::dn, ldap_entry::dnufn, ldap_entry::first_attr, free_entry(), free_rdn(), free_rdnarr(), ldap_getdn(), ldap_rdn::name, ldap_rdn::next, objalloc(), objlock(), objunlock(), objunref(), ldap_rdn::prev, ldap_entry::rdn, ldap_entry::rdncnt, and ldap_rdn::value.

Referenced by dts_ldapsearch().

```
                                {
LDAPMessage *message = *msgptr;
struct ldap_entry *ent = NULL;
struct ldap_rdn *lrdn, *prev = NULL, *first = NULL;
struct ldap_rdn **rdns;
LDAPDN dnarr;
LDAPRDN rdnarr;
LDAPAVA *rdn;
int res, cnt, tlen=0, dccnt=0;

objlock(ld);
if (message) {
    message = ldap_next_entry(ld, message);
} else {
    message = ldap_first_entry(ld, result);
}
*msgptr = message;
objunlock(ld);

if (message && !(ent = objalloc(sizeof(*ent), free_entry)
  )) {
    if (!err) {
        *err = LDAP_NO_MEMORY;
    }
    return NULL;
} else
    if (!message) {
        if (err) {
            objlock(ld);
            ldap_get_option(ld, LDAP_OPT_RESULT_CODE, err);
            objunlock(ld);
        }
        return NULL;
    }

if (!(ent->dn = ldap_getdn(ld, message, &res))) {
    if (err) {
        *err = res;
    }
    objunref(ent);
    return NULL;
}

objlock(ld);
if ((res = ldap_str2dn(ent->dn, &dnarr, LDAP_DN_PEDANTIC))) {
    objunlock(ld);
    if (err) {
        *err = res;
    }
    objunref(ent);
    return NULL;
}
objunlock(ld);

ent->rdncnt = 0;
for (cnt=0; dnarr[cnt]; cnt++) {
    rdnarr = dnarr[cnt];
    for (; *rdnarr; rdnarr++) {
        if (!(lrdn = objalloc(sizeof(*lrdn), free_rdn))) {
            for(lrdn = first; lrdn; lrdn=lrdn->next) {
                objunref(lrdn);
            }
            objunref(ent);
            if (err) {
                *err = LDAP_NO_MEMORY;
            }
            return NULL;
        }

        ent->rdncnt++;

        if (!first) {
            first = lrdn;
        }

        rdn = *rdnarr;
        ALLOC_CONST(lrdn->name, rdn->la_attr.bv_val);
        ALLOC_CONST(lrdn->value, rdn->la_value.bv_val);

        if (!strcmp("dc", rdn->la_attr.bv_val)) {
            dccnt++;
        }
        tlen += rdn->la_value.bv_len;
        lrdn->next = NULL;
        if (prev) {
            prev->next = lrdn;
```

```
                lrdn->prev = prev;
            } else {
                lrdn->prev = NULL;
            }
            prev = lrdn;
        }
    }
    ldap_dnfree(dnarr);

    ent->dnufn = calloc(tlen + (ent->rdncnt-dccnt)*2+dccnt, 1);
    ent->rdn = rdns = objalloc(sizeof(void *) * (ent->rdncnt+1
      ), free_rdnarr);

    if (!ent->dnufn || !ent->rdn) {
        for(lrdn = first; lrdn; lrdn=lrdn->next) {
            objunref(lrdn);
        }
        objunref(ent);
        if (err) {
            *err = LDAP_NO_MEMORY;
        }
    }

    for(lrdn = first; lrdn ; lrdn = lrdn->next) {
        strcat((char *)ent->dnufn, lrdn->value);
        if (lrdn->next && !strcmp(lrdn->name, "dc")) {
            strcat((char *)ent->dnufn, ".");
        } else
            if (lrdn->next) {
                strcat((char *)ent->dnufn, ", ");
            }
        *rdns = lrdn;
        rdns++;
    }
    *rdns = NULL;

    if (!(ent->attrs = attr2bl(ld, message, &ent->first_attr
      , b64enc, &res))) {
        if (err) {
            *err = res;
        }
        objunref(ent);
        return NULL;
    }

    if (err) {
        *err = LDAP_SUCCESS;
    }

    return ent;
}
```

7.36.1.36 struct ldap_entry* ldap_getentry (struct ldap_results * *results*, const char * *dn*) [read]

Definition at line 1083 of file openldap.c.

References bucket_list_find_key(), and ldap_results::entries.

```
          {
    if (!results || !dn) {
        return NULL;
    }
    return (struct ldap_entry *)bucket_list_find_key
      (results->entries, dn);
}
```

7.36.1.37 int ldap_mod_add (struct ldap_modify * *lmod*, const char * *attr*, ...)

Definition at line 1207 of file openldap.c.

References add_modifyval(), getmodreq(), and objunref().

Referenced by ldap_mod_addattr().

```
{
```

```
    va_list a_list;
    char *val;
    struct ldap_modreq *modr;

    if (!(modr = getmodreq(lmod, attr, LDAP_MOD_ADD))) {
        return 1;
    }

    va_start(a_list, attr);
    while((val = va_arg(a_list, void *))) {
        if (add_modifyval(modr, val)) {
            objunref(modr);
            return(1);
        }
    }

    objunref(modr);
    va_end(a_list);
    return 0;
}
```

7.36.1.38 int ldap_mod_addattr (struct ldap_conn * *ldap*, const char * *dn*, const char * *attr*, const char * *value*)

Definition at line 1359 of file openldap.c.

References ldap_domodify(), ldap_mod_add(), ldap_modifyinit(), and objunref().

```
                                {
    int res = 0;
    struct ldap_modify *lmod;

    if (!(lmod = ldap_modifyinit(dn))) {
        return LDAP_NO_MEMORY;
    }

    if (ldap_mod_add(lmod, attr, value, NULL)) {
        objunref(lmod);
        return LDAP_NO_MEMORY;
    }

    res = ldap_domodify(ldap, lmod);
    objunref(lmod);
    return res;
}
```

7.36.1.39 int ldap_mod_del (struct ldap_modify * *lmod*, const char * *attr*, ...)

Definition at line 1185 of file openldap.c.

References add_modifyval(), getmodreq(), and objunref().

Referenced by ldap_mod_delattr().

```
                                                                        {
    va_list a_list;
    char *val;
    struct ldap_modreq *modr;

    if (!(modr = getmodreq(lmod, attr, LDAP_MOD_DELETE))) {
        return 1;
    }

    va_start(a_list, attr);
    while((val = va_arg(a_list, void *))) {
        if (add_modifyval(modr, val)) {
            objunref(modr);
            return(1);
        }
    }

    objunref(modr);
    va_end(a_list);
    return 0;
}
```

7.36.1.40 int ldap_mod_delattr (struct ldap_conn * *ldap,* const char * *dn,* const char * *attr,* const char * *value*)

Definition at line 1338 of file openldap.c.

References ldap_domodify(), ldap_mod_del(), ldap_modifyinit(), and objunref().

Referenced by ldap_mod_remattr().

```
                                {
    struct ldap_modify *lmod;
    int res;

    if (!(lmod = ldap_modifyinit(dn))) {
        return LDAP_NO_MEMORY;
    }
    if (ldap_mod_del(lmod, attr, value, NULL)) {
        objunref(lmod);
        return LDAP_NO_MEMORY;
    }

    res = ldap_domodify(ldap, lmod);
    objunref(lmod);
    return res;
}
```

7.36.1.41 int ldap_mod_remattr (struct ldap_conn * *ldap,* const char * *dn,* const char * *attr*)

Definition at line 1355 of file openldap.c.

References ldap_mod_delattr().

```
            {
    return ldap_mod_delattr(ldap, dn, attr, NULL);
}
```

7.36.1.42 int ldap_mod_rep (struct ldap_modify * *lmod,* const char * *attr,* ...)

Definition at line 1229 of file openldap.c.

References add_modifyval(), getmodreq(), and objunref().

Referenced by ldap_mod_repattr().

```
                                                                        {
    va_list a_list;
    char *val;
    struct ldap_modreq *modr;

    if (!(modr = getmodreq(lmod, attr, LDAP_MOD_REPLACE))) {
        return 1;
    }

    va_start(a_list, attr);
    while((val = va_arg(a_list, void *))) {
        if (add_modifyval(modr, val)) {
            objunref(modr);
            return(1);
        }
    }

    objunref(modr);
    va_end(a_list);
    return 0;
}
```

7.36.1.43 int ldap_mod_repattr (struct ldap_conn * *ldap,* const char * *dn,* const char * *attr,* const char * *value*)

Definition at line 1377 of file openldap.c.

References ldap_domodify(), ldap_mod_rep(), ldap_modifyinit(), and objunref().

```
                                {
    struct ldap_modify *lmod;
    int res;

    if (!(lmod = ldap_modifyinit(dn))) {
        return LDAP_NO_MEMORY;
    }

    if (ldap_mod_rep(lmod, attr, value, NULL)) {
        objunref(lmod);
        return LDAP_NO_MEMORY;
    }

    res = ldap_domodify(ldap, lmod);
    objunref(lmod);
    return res;
}
```

7.36.1.44 struct ldap_modify* ldap_modifyinit (const char * *dn*) [read]

Definition at line 1097 of file openldap.c.

References ALLOC_CONST, ldap_modify::bl, create_bucketlist(), ldap_modify::dn, free_modify(), modify_hash(), objalloc(), and objunref().

Referenced by ldap_mod_addattr(), ldap_mod_delattr(), and ldap_mod_repattr().

```
                                                                {
    struct ldap_modify *mod;
    int cnt;

    if (!(mod = objalloc(sizeof(*mod), free_modify))) {
        return NULL;
    }

    ALLOC_CONST(mod->dn, dn);
    if (!mod->dn) {
        objunref(mod);
        return NULL;
    }

    for(cnt=0; cnt < 3; cnt++) {
        if (!(mod->bl[cnt] = create_bucketlist(4,
      modify_hash))) {
            objunref(mod);
            return NULL;
        }
    }

    return mod;
}
```

7.36.1.45 int ldap_rebind_proc (LDAP * *ld,* LDAP_CONST char * *url,* ber_tag_t *request,* ber_int_t *msgid,* void * *params*)

Definition at line 271 of file openldap.c.

References ldap_simple::cred, ldap_simple::dn, dts_sasl_interact(), ldap_conn::ldap, sasl_defaults::mech, objref(), objunref(), ldap_conn::sasl, ldap_conn::sctrlsp, and ldap_conn::simple.

Referenced by ldap_connect().

```
                            {
    struct ldap_conn *ldap = params;
    int res = LDAP_UNAVAILABLE;

    if (!objref(ldap)) {
        return LDAP_UNAVAILABLE;
    }

    if (ldap->sasl) {
        int sasl_flags = LDAP_SASL_AUTOMATIC | LDAP_SASL_QUIET;
        struct sasl_defaults *sasl = ldap->sasl;
```

```
        res = ldap_sasl_interactive_bind_s(ld, NULL, sasl->mech, ldap->
      sctrlsp , NULL, sasl_flags, dts_sasl_interact, sasl);
    } else
        if (ldap->simple) {
            struct ldap_simple *simple = ldap->simple;

            res = ldap_sasl_bind_s(ld, simple->dn, LDAP_SASL_SIMPLE, simple->
      cred, ldap->sctrlsp, NULL, NULL);
        }

    objunref(ldap);
    return res;
}
```

7.36.1.46 LDAPMod* ldap_reqtoarr (struct ldap_modreq * *modr,* int *type*)

Definition at line 1251 of file openldap.c.

References ldap_modreq::attr, ldap_modreq::cnt, ldap_modreq::first, ldap_modval::next, and ldap_modval::value.

Referenced by ldap_doadd(), and ldap_domodify().

```
                                                                {
    LDAPMod *modi;
    const char **mval;
    struct ldap_modval *modv;

    if (!(modi = calloc(sizeof(LDAPMod), 1))) {
        return NULL;
    }

    if (!(modi->mod_values = calloc(sizeof(void *), modr->cnt+1))) {
        free(modi);
        return NULL;
    }

    switch (type) {
        case 0:
            modi->mod_op = LDAP_MOD_REPLACE;
            break;
        case 1:
            modi->mod_op = LDAP_MOD_DELETE;
            break;
        case 2:
            modi->mod_op = LDAP_MOD_ADD;
            break;
        default
                :
            modi->mod_op = 0;
            break;
    }

    if (!(modi->mod_type = strdup(modr->attr))) {
        free(modi);
        return NULL;
    }

    mval = (const char **)modi->mod_values;
    for(modv = modr->first; modv; modv=modv->next) {
        if (!(*mval = strdup(modv->value))) {
            ldap_mods_free(&modi, 0);
            return NULL;
        }
        mval++;
    }
    *mval = NULL;

    return modi;
}
```

7.36.1.47 int ldap_saslbind (struct ldap_conn * *ld,* const char * *mech,* const char * *realm,* const char * *authcid,* const char * *passwd,* const char * *authzid*)

Definition at line 458 of file openldap.c.

References ALLOC_CONST, sasl_defaults::authcid, sasl_defaults::authzid, dts_sasl_interact(), free_sasl(), ldap_conn::ldap, sasl_defaults::mech, objalloc(), objlock(), objref(), objunlock(), objunref(), sasl_defaults::passwd, sasl_defaults::realm, ldap_conn::sasl, and ldap_conn::sctrlsp.

```
                                                                    {
    struct sasl_defaults *sasl;
    int res, sasl_flags = LDAP_SASL_AUTOMATIC | LDAP_SASL_QUIET;

    if (!objref(ld)) {
        return LDAP_UNAVAILABLE;
    }

    if (!(sasl = objalloc(sizeof(*sasl), free_sasl))) {
        return LDAP_NO_MEMORY;
    }

    ALLOC_CONST(sasl->passwd, passwd);

    if (mech) {
        ALLOC_CONST(sasl->mech, mech);
    } else {
        ldap_get_option(ld->ldap, LDAP_OPT_X_SASL_MECH, &sasl->mech);
    }

    if (realm) {
        ALLOC_CONST(sasl->realm, realm);
    } else {
        ldap_get_option(ld->ldap, LDAP_OPT_X_SASL_REALM, &sasl->realm
      );
    }

    if (authcid) {
        ALLOC_CONST(sasl->authcid, authcid);
    } else {
        ldap_get_option(ld->ldap, LDAP_OPT_X_SASL_AUTHCID, &sasl->authcid
       );
    }

    if (authzid) {
        ALLOC_CONST(sasl->authzid, authzid);
    } else {
        ldap_get_option(ld->ldap, LDAP_OPT_X_SASL_AUTHZID, &sasl->authzid
       );
    }

    objlock(ld);
    if (ld->sasl) {
        objunref(ld->sasl);
    }
    ld->sasl = sasl;
    res = ldap_sasl_interactive_bind_s(ld->ldap, NULL, sasl->mech, ld->
      sctrlsp , NULL, sasl_flags, dts_sasl_interact, sasl);
    objunlock(ld);
    objunref(ld);
    return res;
}
```

7.36.1.48 struct ldap_results* ldap_search_base (struct ldap_conn * *ld*, const char * *base*, const char * *filter*, int *b64enc*, int * *res*, ...) [read]

Definition at line 667 of file openldap.c.

References dts_ldapsearch(), and malloc.

```
                                                                    {
    va_list a_list;
    char *attr, **tmp, **attrs = NULL;
    int cnt = 1;

    va_start(a_list, res);
    while (( attr=va_arg(a_list, void *))) {
        cnt++;
    }
    va_end(a_list);

    if (cnt > 1) {
        tmp = attrs = malloc(sizeof(void *)*cnt);

        va_start(a_list, res);
```

```
        while (( attr=va_arg(a_list, char *))) {
            *tmp = attr;
            tmp++;
        }
        va_end(a_list);
        *tmp=NULL;
    }

    return dts_ldapsearch(ld, base, LDAP_SCOPE_BASE, filter,
      attrs, b64enc, res);
}
```

7.36.1.49 struct ldap_results* ldap_search_one (struct ldap_conn * *ld,* const char * *base,* const char * *filter,* int *b64enc,* int * *res,* ...) [read]

Definition at line 641 of file openldap.c.

References dts_ldapsearch(), and malloc.

```
{
    va_list a_list;
    char *attr, **tmp, **attrs = NULL;
    int cnt = 1;

    va_start(a_list, res);
    while (( attr=va_arg(a_list, void *))) {
        cnt++;
    }
    va_end(a_list);

    if (cnt > 1) {
        tmp = attrs = malloc(sizeof(void *)*cnt);

        va_start(a_list, res);
        while (( attr=va_arg(a_list, char *))) {
            *tmp = attr;
            tmp++;
        }
        va_end(a_list);
        *tmp=NULL;
    }

    return dts_ldapsearch(ld, base, LDAP_SCOPE_ONELEVEL, filter,
      attrs, b64enc, res);
}
```

7.36.1.50 struct ldap_results* ldap_search_sub (struct ldap_conn * *ld,* const char * *base,* const char * *filter,* int *b64enc,* int * *res,* ...) [read]

Definition at line 615 of file openldap.c.

References dts_ldapsearch(), and malloc.

Referenced by ldap_simplerebind().

```
{
    va_list a_list;
    char *attr, **tmp, **attrs = NULL;
    int cnt = 1;

    va_start(a_list, res);
    while (( attr=va_arg(a_list, void *))) {
        cnt++;
    }
    va_end(a_list);

    if (cnt > 1) {
        tmp = attrs = malloc(sizeof(void *)*cnt);

        va_start(a_list, res);
        while (( attr=va_arg(a_list, char *))) {
            *tmp = attr;
            tmp++;
```

```
        }
        va_end(a_list);
        *tmp=NULL;
    }

    return dts_ldapsearch(ld, base, LDAP_SCOPE_SUBTREE, filter,
      attrs, b64enc, res);
}
```

7.36.1.51 int ldap_simplebind (struct ldap_conn * *ld,* const char * *dn,* const char * *passwd*)

Definition at line 389 of file openldap.c.

References ldap_simple::cred, ldap_simple::dn, free_simple(), ldap_conn::ldap, malloc, objalloc(), objlock(), objref(), objunlock(), objunref(), ldap_conn::sctrlsp, and ldap_conn::simple.

Referenced by ldap_simplerebind().

```
        {
    struct ldap_simple *simple;
    struct berval *cred;
    int res, len = 0;

    if (!objref(ld)) {
        return LDAP_UNAVAILABLE;
    }

    if (passwd) {
        len = strlen(passwd);
    }
    simple = objalloc(sizeof(*simple), free_simple);
    cred = calloc(sizeof(*cred), 1);
    cred->bv_val = malloc(len);
    memcpy(cred->bv_val, passwd, len);
    cred->bv_len=len;
    simple->cred = cred;
    simple->dn = strdup(dn);

    objlock(ld);
    if (ld->simple) {
        objunref(ld->simple);
    }
    ld->simple = simple;
    res = ldap_sasl_bind_s(ld->ldap, simple->dn, LDAP_SASL_SIMPLE, simple
      ->cred, ld->sctrlsp, NULL, NULL);
    objunlock(ld);
    objunref(ld);
    return res;
}
```

7.36.1.52 int ldap_simplerebind (struct ldap_conn * *ldap,* const char * *initialdn,* const char * *initialpw,* const char * *base,* const char * *filter,* const char * *uidrdn,* const char * *uid,* const char * *passwd*)

Definition at line 420 of file openldap.c.

References ldap_results::count, ldap_entry::dn, ldap_results::first_entry, ldap_search_sub(), ldap_simplebind(), malloc, objref(), and objunref().

```
        {
    int res, flen;
    struct ldap_results *results;
    const char *sfilt;

    if (!objref(ldap)) {
        return LDAP_UNAVAILABLE;
    }

    if ((res = ldap_simplebind(ldap, initialdn, initialpw))) {
        objunref(ldap);
        return res;
    }

    flen=strlen(uidrdn) + strlen(filter) + strlen(uid) + 7;
```

```
    sfilt = malloc(flen);
    snprintf((char *)sfilt, flen, "(&(%s=%s)%s)", uidrdn, uid, filter);

    if (!(results = ldap_search_sub(ldap, base, sfilt, 0, &res,
      uidrdn, NULL))) {
        free((void *)sfilt);
        objunref(ldap);
        return res;
    }
    free((void *)sfilt);

    if (results->count != 1) {
        objunref(results);
        objunref(ldap);
        return LDAP_INAPPROPRIATE_AUTH;
    }

    res = ldap_simplebind(ldap, results->first_entry
      ->dn, passwd);
    objunref(ldap);
    objunref(results);
    return res;
}
```

7.36.1.53 void ldap_unref_attr (struct ldap_entry * *entry,* struct ldap_attr * *attr*)

Definition at line 1053 of file openldap.c.

References ldap_entry::attrs, ldap_entry::first_attr, ldap_attr::next, objcnt(), objunref(), and remove_bucket_item().

```
                                                                    {
    if (!entry || !attr) {
        return;
    }

    if (objcnt(attr) > 1) {
        objunref(attr);
    } else {
        if (attr == entry->first_attr) {
            entry->first_attr = attr->next;
        }
        remove_bucket_item(entry->attrs, attr);
    }
}
```

7.36.1.54 void ldap_unref_entry (struct ldap_results * *results,* struct ldap_entry * *entry*)

Definition at line 1068 of file openldap.c.

References ldap_results::entries, ldap_results::first_entry, ldap_entry::next, objcnt(), objunref(), and remove_bucket_item().

```
            {
    if (!results || !entry) {
        return;
    }

    if (objcnt(entry) > 1) {
        objunref(entry);
    } else {
        if (entry == results->first_entry) {
            results->first_entry = entry->next;
        }
        remove_bucket_item(results->entries, entry);
    }
}
```

7.36.1.55 int ldapattr_hash (const void * *data,* int *key*)

Definition at line 814 of file openldap.c.

References jenhash, and ldap_attr::name.

Referenced by attr2bl().

```
                                                {
    int ret;
    const struct ldap_attr *la = data;
    const char *hashkey = (key) ? data : la->name;

    if (hashkey) {
        ret = jenhash(hashkey, strlen(hashkey), 0);
    } else {
        ret = jenhash(la, sizeof(la), 0);
    }
    return(ret);
}
```

7.36.1.56 int modify_hash (const void * *data*, int *key*)

Definition at line 258 of file openldap.c.

References ldap_modreq::attr, and jenhash.

Referenced by ldap_addinit(), and ldap_modifyinit().

```
                                             {
    int ret;
    const struct ldap_modreq *modr = data;
    const char *hashkey = (key) ? data : modr->attr;

    if (hashkey) {
        ret = jenhash(hashkey, strlen(hashkey), 0);
    } else {
        ret = jenhash(modr, sizeof(modr), 0);
    }
    return(ret);
}
```

7.36.1.57 struct ldap_modreq* new_modreq (struct bucket_list * *modtype*, const char * *attr*) [read]

Definition at line 1121 of file openldap.c.

References addtobucket(), ALLOC_CONST, ldap_modreq::attr, free_modreq(), objalloc(), and objunref().

Referenced by getaddreq(), and getmodreq().

```
                                                                        {
    struct ldap_modreq *modr;

    if (!(modr = objalloc(sizeof(*modr), free_modreq))) {
        return NULL;
    }

    ALLOC_CONST(modr->attr, attr);
    if (!modr->attr || !addtobucket(modtype, modr)) {
        objunref(modr);
        modr = NULL;
    }
    return modr;
}
```

7.36.1.58 int searchresults_hash (const void * *data*, int *key*)

Definition at line 516 of file openldap.c.

References ldap_entry::dn, and jenhash.

Referenced by dts_ldapsearch().

```
                                                {
    int ret;
    const struct ldap_entry *ent = data;
    const char *hashkey = (key) ? data : ent->dn;

    if (hashkey) {
        ret = jenhash(hashkey, strlen(hashkey), 0);
    } else {
        ret = jenhash(ent, sizeof(ent), 0);
    }
    return(ret);
}
```

7.37 src/radius.c File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>
#include <uuid/uuid.h>
#include <openssl/md5.h>
#include "include/dtsapp.h"
```

Data Structures

- struct radius_packet
- struct radius_session
- struct radius_connection
- struct radius_server

Functions

- struct radius_connection * radconnect (struct radius_server *server)
- unsigned char * addradattr (struct radius_packet *packet, char type, unsigned char *val, char len)
- void addradattrint (struct radius_packet *packet, char type, unsigned int val)
- void addradattrip (struct radius_packet *packet, char type, char *ipaddr)
- void addradattrstr (struct radius_packet *packet, char type, char *str)
- struct radius_packet * new_radpacket (unsigned char code, unsigned char id)
- void add_radserver (const char *ipaddr, const char *auth, const char *acct, const char *secret, int timeout)
- int send_radpacket (struct radius_packet *packet, const char *userpass, radius_cb read_cb, void *cb_data)
- unsigned char * radius_attr_first (struct radius_packet *packet)
- unsigned char * radius_attr_next (struct radius_packet *packet, unsigned char *attr)

7.37.1 Function Documentation

7.37.1.1 void add_radserver (const char * *ipaddr*, const char * *auth*, const char * *acct*, const char * *secret*, int *timeout*)

Definition at line 233 of file radius.c.

References radius_server::acctport, addtobucket(), ALLOC_CONST, radius_server::authport, bucket_list_cnt(), create_bucketlist(), radius_server::id, radius_server::name, objalloc(), objunref(), radius_server::secret, radius_server::service, and radius_server::timeout.

```
                                        {
    struct radius_server *server;

    if ((server = objalloc(sizeof(*server), del_radserver))) {
        ALLOC_CONST(server->name, ipaddr);
        ALLOC_CONST(server->authport, auth);
        ALLOC_CONST(server->acctport, acct);
        ALLOC_CONST(server->secret, secret);
        if (!servers) {
            servers = create_bucketlist(0, hash_server);
        }
        server->id = bucket_list_cnt(servers);
        server->timeout = timeout;
        gettimeofday(&server->service, NULL);
        addtobucket(servers, server);
    }

    objunref(server);
}
```

7.37.1.2 unsigned char* addradattr (struct radius_packet * *packet,* char *type,* unsigned char * *val,* char *len*)

Definition at line 95 of file radius.c.

References radius_packet::attrs, radius_packet::len, and RAD_AUTH_HDR_LEN.

Referenced by addradattrint(), addradattrip(), and addradattrstr().

```
                                {
    unsigned char *data = packet->attrs + packet->len -
      RAD_AUTH_HDR_LEN;

    if (!len) {
        return NULL;
    }

    data[0] = type;
    data[1] = len + 2;
    if (val) {
        memcpy(data + 2, val, len);
    }

    packet->len += data[1];
    return (data);
}
```

7.37.1.3 void addradattrint (struct radius_packet * *packet,* char *type,* unsigned int *val*)

Definition at line 112 of file radius.c.

References addradattr().

```
            {
    unsigned int tval;

    tval = htonl(val);
    addradattr(packet, type, (unsigned char *)&tval, sizeof(tval));
}
```

7.37.1.4 void addradattrip (struct radius_packet * *packet,* char *type,* char * *ipaddr*)

Definition at line 119 of file radius.c.

References addradattr().

```
       {
    unsigned int tval;
```

```
    tval = inet_addr(ipaddr);
    addradattr(packet, type, (unsigned char *)&tval, sizeof(tval));
}
```

7.37.1.5 void addradattrstr (struct radius_packet * *packet,* char *type,* char * *str*)

Definition at line 126 of file radius.c.

References addradattr().

```
                                                                    {
    addradattr(packet, type, (unsigned char *)str, strlen(str));
}
```

7.37.1.6 struct radius_packet* new_radpacket (unsigned char *code,* unsigned char *id*) [read]

Definition at line 171 of file radius.c.

References radius_packet::code, genrand(), radius_packet::len, malloc, RAD_AUTH_HDR_LEN, RAD_AUTH_TOKEN_LEN, and radius_packet::token.

```
        {
    struct radius_packet *packet;

    if ((packet = malloc(sizeof(*packet)))) {
        memset(packet, 0, sizeof(*packet));
        packet->len = RAD_AUTH_HDR_LEN;
        packet->code = code;
        genrand(&packet->token, RAD_AUTH_TOKEN_LEN
      );
    }
    return (packet);
}
```

7.37.1.7 struct radius_connection * radconnect (struct radius_server * *server*) [read]

Definition at line 541 of file radius.c.

References addtobucket(), radius_server::authport, radius_server::connex, create_bucketlist(), framework_mkthread(), genrand(), radius_connection::id, radius_server::name, objalloc(), radius_connection::server, fwsocket::sock, radius_connection::socket, and udpconnect().

```
                                                                    {
    struct radius_connection *connex;
    int val = 1;

    if ((connex = objalloc(sizeof(*connex), del_radconnect))) {
        if ((connex->socket = udpconnect(server->name,
      server->authport, NULL))) {
            if (!server->connex) {
                server->connex = create_bucketlist(0,
      hash_connex);
            }
            setsockopt(connex->socket->sock, SOL_IP, IP_RECVERR,(char
       *)&val, sizeof(val));
            connex->server = server;
            genrand(&connex->id, sizeof(connex->id));
            addtobucket(server->connex, connex);
            framework_mkthread(rad_return, NULL, NULL, connex
      );
        }
    }
    return (connex);
}
```

7.37.1.8 unsigned char* radius_attr_first (struct radius_packet * *packet*)

Definition at line 560 of file radius.c.

References radius_packet::attrs.

```
                                                                    {
    return (packet->attrs);
}
```

7.37.1.9 unsigned char* radius_attr_next (struct radius_packet * *packet,* unsigned char * *attr*)

Definition at line 564 of file radius.c.

References radius_packet::attrs, radius_packet::len, and RAD_AUTH_HDR_LEN.

```
                {
    int offset = (packet->len - RAD_AUTH_HDR_LEN) - (attr -
      packet->attrs);

    if (!(offset - attr[1])) {
        return NULL;
    }

    return (attr + attr[1]);
}
```

7.37.1.10 int send_radpacket (struct radius_packet * *packet,* const char * *userpass,* radius_cb *read_cb,* void * *cb_data*)

Definition at line 390 of file radius.c.

```
                                        {
    return (_send_radpacket(packet, userpass, NULL, read_cb, cb_data));
}
```

7.38 src/refobj.c File Reference

Referenced Objects.

```
#include <pthread.h>
#include <string.h>
#include <stdlib.h>
#include <stdint.h>
#include "include/dtsapp.h"
```

Data Structures

- struct ref_obj
- struct blist_obj
- struct bucket_list
- struct bucket_loop

Macros

- #define REFOBJ_MAGIC 0xdeadc0de
- #define refobj_offset sizeof(struct ref_obj);

Functions

- void * objalloc (int size, objdestroy destructor)
- int objref (void *data)
- int objunref (void *data)
- int objcnt (void *data)
- int objsize (void *data)
- int objlock (void *data)
- int objtrylock (void *data)
- int objunlock (void *data)
- void * create_bucketlist (int bitmask, blisthash hash_function)
- int addtobucket (struct bucket_list *blist, void *data)
- struct bucket_loop * init_bucket_loop (struct bucket_list *blist)
- void stop_bucket_loop (struct bucket_loop *bloop)
- void * next_bucket_loop (struct bucket_loop *bloop)
- void remove_bucket_item (struct bucket_list *blist, void *data)
- void remove_bucket_loop (struct bucket_loop *bloop)
- int bucket_list_cnt (struct bucket_list *blist)
- void * bucket_list_find_key (struct bucket_list *blist, const void *key)
- void bucketlist_callback (struct bucket_list *blist, blist_cb callback, void *data2)
- void * objchar (const char *orig)

7.38.1 Detailed Description

Referenced Objects.

Definition in file refobj.c.

7.39 src/rfc6296.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include "include/dtsapp.h"
```

Data Structures

- struct natmap

Functions

- void rfc6296_map (struct natmap *map, struct in6_addr *ipaddr, int out)
- int rfc6296_map_add (char *intaddr, char *extaddr)
- void rfc6296_test (blist_cb callback, struct in6_addr *internal)

Variables

- struct bucket_list * nptv6tbl = NULL

7.39.1 Function Documentation

7.39.1.1 void rfc6296_map (struct natmap * *map,* struct in6_addr * *ipaddr,* int *out*)

Definition at line 47 of file rfc6296.c.

References natmap::adji, natmap::adjo, natmap::epre, natmap::ipre, and natmap::mask.

```
                                                                                {
    uint16_t *addr_16 = (uint16_t *)&ipaddr->s6_addr;
    uint32_t calc;
    uint8_t cnt, *prefix, bitlen, bytelen;
    uint16_t adj;

    prefix = (out) ? map->epre : map->ipre;
    adj = (out) ? map->adjo : map->adji;

    if ((bitlen = map->mask % 8)) {
        bytelen = (map->mask - bitlen) / 8;
        bytelen++;
    } else {
        bytelen = map->mask / 8;
    }

    /*as per RFC we handle /48 and longer /48 changes are reflected in SN*/
    if ((bytelen == 6) && (~addr_16[3]) && (!bitlen)) {
        memcpy(&ipaddr->s6_addr, prefix, bytelen);
        calc = ntohs(addr_16[3]) + adj;
        addr_16[3] = htons((calc & 0xFFFF) + (calc >> 16));
        if (! ~addr_16[3]) {
            addr_16[3] = 0;
        }
    } else
        if ((bytelen > 6) && (bytelen < 15)) {
            /* find first non 0xFFFF word in lower 64 bits*/
            for(cnt = ((bytelen-1) >> 1) + 1; cnt < 8; cnt++) {
                if (! ~addr_16[cnt]) {
                    continue;
                }
                if (bitlen) {
                    ipaddr->s6_addr[bytelen-1] = prefix[bytelen-1] | (ipaddr->
      s6_addr[bytelen-1] & ((1 << (8 - bitlen)) -1));
                } else {
                    ipaddr->s6_addr[bytelen-1] = prefix[bytelen-1];
                }
                memcpy(&ipaddr->s6_addr, prefix, bytelen - 1);
                calc = ntohs(addr_16[cnt]) + adj;
                addr_16[cnt] = htons((calc & 0xFFFF) + (calc >> 16));
                if (! ~addr_16[cnt]) {
                    addr_16[cnt] = 0;
                }
                break;
            }
        }
}
```

7.39.1.2 int rfc6296_map_add (char * *intaddr,* char * *extaddr*)

Definition at line 94 of file rfc6296.c.

References addtobucket(), natmap::adji, natmap::adjo, checksum(), create_bucketlist(), natmap::epre, natmap::ipre, natmap::mask, objalloc(), and objunref().

```
                                                          {
    struct natmap *map;
    uint16_t emask, imask, isum, esum, bytelen, bitlen;
    char inip[43], exip[43], *tmp2;
    struct in6_addr i6addr;
    uint32_t adj;

    strncpy(inip, intaddr, 43);
    if ((tmp2 = rindex(inip, '/'))) {
        tmp2[0] = '\0';
        tmp2++;
        imask = atoi(tmp2);
    } else {
        return (-1);
    }
```

```
    strncpy(exip, extaddr, 43);
    if ((tmp2 = rindex(exip, '/'))) {
        tmp2[0] = '\0';
        tmp2++;
        emask = atoi(tmp2);
    } else {
        return (-1);
    }

    map = objalloc(sizeof(*map), NULL);
    map->mask = (emask > imask) ? emask : imask;

    /*rfc says we must zero extend this is what we do here looking at each
       supplied len*/
    /*external range*/
    inet_pton(AF_INET6, exip, &i6addr);
    if ((bitlen = emask % 8)) {
        bytelen = (emask - bitlen) / 8;
        i6addr.s6_addr[bytelen] &= ~((1 << (8 - bitlen)) - 1);
        bytelen++;
    } else {
        bytelen = emask / 8;
    }
    memcpy(map->epre, &i6addr.s6_addr, bytelen);

    /*internal range*/
    inet_pton(AF_INET6, inip, &i6addr);
    if ((bitlen = imask % 8)) {
        bytelen = (imask - bitlen) / 8;
        i6addr.s6_addr[bytelen] &= ~((1 << (8 - bitlen)) - 1);
        bytelen++;
    } else {
        bytelen = imask / 8;
    }
    memcpy(map->ipre, &i6addr.s6_addr, bytelen);

    /*calculate the adjustments from checksums of prefixes*/
    if ((bitlen = map->mask % 8)) {
        bytelen = (map->mask - bitlen) / 8;
        bytelen++;
    } else {
        bytelen = map->mask / 8;
    }
    esum = ntohs(checksum(map->epre, bytelen));
    isum = ntohs(checksum(map->ipre, bytelen));

    /*outgoing transform*/
    adj = esum - isum;
    adj = (adj & 0xFFFF) + (adj >> 16);
    map->adjo = (uint16_t)adj;

    /*incoming transform*/
    adj = isum - esum;
    adj = (adj & 0xFFFF) + (adj >> 16);
    map->adji = (uint16_t)adj;

    if (!nptv6tbl && (!(nptv6tbl = create_bucketlist
      (5, nptv6_hash)))) {
        objunref(map);
        return (-1);
    }
    addtobucket(nptv6tbl, map);
    objunref(map);

    return (0);
}
```

7.39.1.3 void rfc6296_test (blist_cb *callback*, struct in6_addr * *internal*)

Definition at line 175 of file rfc6296.c.

References bucketlist_callback(), and objunref().

```
                                                                    {
    /*find and run map*/
    bucketlist_callback(nptv6tbl, callback, internal
      );

    objunref(nptv6tbl);
}
```

7.39.2 Variable Documentation

7.39.2.1 struct bucket_list* nptv6tbl = NULL

Definition at line 35 of file rfc6296.c.

7.40 src/socket.c File Reference

```
#include <netdb.h>
#include <unistd.h>
#include <stdint.h>
#include <string.h>
#include <errno.h>
#include <stdio.h>
#include <fcntl.h>
#include <arpa/inet.h>
#include "include/dtsapp.h"
#include "include/private.h"
```

Data Structures

- struct socket_handler

Functions

- void close_socket (struct fwsocket *sock)
- struct fwsocket * make_socket (int family, int type, int proto, void *ssl)
- struct fwsocket * sockconnect (int family, int stype, int proto, const char *ipaddr, const char *port, void *ssl)
- struct fwsocket * udpconnect (const char *ipaddr, const char *port, void *ssl)
- struct fwsocket * tcpconnect (const char *ipaddr, const char *port, void *ssl)
- struct fwsocket * sockbind (int family, int stype, int proto, const char *ipaddr, const char *port, void *ssl, int backlog)
- struct fwsocket * udpbind (const char *ipaddr, const char *port, void *ssl)
- struct fwsocket * tcpbind (const char *ipaddr, const char *port, void *ssl, int backlog)
- void socketserver (struct fwsocket *sock, socketrecv read, socketrecv acceptfunc, threadcleanup cleanup, void *data)
- void socketclient (struct fwsocket *sock, void *data, socketrecv read, threadcleanup cleanup)

7.40.1 Function Documentation

7.40.1.1 void close_socket (struct fwsocket * *sock*)

Definition at line 57 of file socket.c.

References objunref(), setflag, and SOCK_FLAG_CLOSE.

```
                                                {
    if (sock) {
        setflag(sock, SOCK_FLAG_CLOSE);
        objunref(sock);
    }
}
```

7.40.1.2 struct fwsocket* make_socket (int *family,* int *type,* int *proto,* void * *ssl*) [read]

Definition at line 89 of file socket.c.

References objalloc(), objunref(), fwsocket::proto, fwsocket::sock, fwsocket::ssl, and fwsocket::type.

Referenced by dtls_listenssl().

```
      {
    struct fwsocket *si;

    if (!(si = objalloc(sizeof(*si),clean_fwsocket))) {
        return NULL;
    }

    if ((si->sock = socket(family, type, proto)) < 0) {
        objunref(si);
        return NULL;
    };

    if (ssl) {
        si->ssl = ssl;
    }
    si->type = type;
    si->proto = proto;

    return (si);
}
```

7.40.1.3 struct fwsocket* sockbind (int *family,* int *stype,* int *proto,* const char * *ipaddr,* const char * *port,* void * *ssl,* int *backlog*) [read]

Definition at line 212 of file socket.c.

```
                                                    {
    return(_opensocket(family, stype, proto, ipaddr, port, ssl, 1,
      backlog));
}
```

7.40.1.4 struct fwsocket* sockconnect (int *family,* int *stype,* int *proto,* const char * *ipaddr,* const char * *port,* void * *ssl*) [read]

Definition at line 200 of file socket.c.

```
                                                {
    return(_opensocket(family, stype, proto, ipaddr, port, ssl, 0, 0));
}
```

7.40.1.5 void socketclient (struct fwsocket * *sock,* void * *data,* socketrecv *read,* threadcleanup *cleanup*)

Definition at line 374 of file socket.c.

References startsslclient().

```
                          {
    startsslclient(sock);

    _start_socket_handler(sock, read, NULL, cleanup, data);
}
```

7.40.1.6 void socketserver (struct fwsocket * *sock,* socketrecv *read,* socketrecv *acceptfunc,* threadcleanup *cleanup,* void * *data*)

Definition at line 354 of file socket.c.

References fwsocket::children, create_bucketlist(), dtsl_serveropts(), fwsocket::flags, objlock(), objunlock(), fwsocket::ssl, and fwsocket::type.

```
          {
    objlock(sock);
    if (sock->flags & SOCK_FLAG_BIND) {
        if (sock->ssl || !(sock->type == SOCK_DGRAM)) {
            sock->children = create_bucketlist(6,
      hash_socket);
        }
        if (sock->ssl && (sock->type == SOCK_DGRAM)) {
            objunlock(sock);
            dtsl_serveropts(sock);
        } else {
            objunlock(sock);
        }
    } else {
        objunlock(sock);
    }
    _start_socket_handler(sock, read, acceptfunc, cleanup, data);
}
```

7.40.1.7 struct fwsocket* tcpbind (const char * *ipaddr,* const char * *port,* void * *ssl,* int *backlog*) [read]

Definition at line 220 of file socket.c.

```
                    {
    return (_opensocket(PF_UNSPEC, SOCK_STREAM, IPPROTO_TCP, ipaddr, port, ssl
      , 1, backlog));
}
```

7.40.1.8 struct fwsocket* tcpconnect (const char * *ipaddr,* const char * *port,* void * *ssl*) [read]

Definition at line 208 of file socket.c.

```
          {
    return (_opensocket(PF_UNSPEC, SOCK_STREAM, IPPROTO_TCP, ipaddr, port, ssl
      , 0, 0));
}
```

7.40.1.9 struct fwsocket* udpbind (const char * *ipaddr,* const char * *port,* void * *ssl*) [read]

Definition at line 216 of file socket.c.

```
        {
    return (_opensocket(PF_UNSPEC, SOCK_DGRAM, IPPROTO_UDP, ipaddr, port, ssl
      , 1, 0));
}
```

7.40.1.10 struct fwsocket* udpconnect (const char * *ipaddr,* const char * *port,* void * *ssl*) [read]

Definition at line 204 of file socket.c.

Referenced by radconnect().

```
        {
    return (_opensocket(PF_UNSPEC, SOCK_DGRAM, IPPROTO_UDP, ipaddr, port, ssl
      , 0, 0));
}
```

7.41 src/sslutil.c File Reference

```
#include <stdint.h>
#include <openssl/ssl.h>
#include <openssl/err.h>
#include <sys/stat.h>
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include "include/dtsapp.h"
```

Data Structures

- struct ssldata

Macros

- #define COOKIE_SECRET_LENGTH 32

Enumerations

- enum SSLFLAGS {
  SSL_TLSV1 = 1 << 0, SSL_SSLV2 = 1 << 1, SSL_SSLV3 = 1 << 2, SSL_DTLSV1 = 1 << 3,
  SSL_CLIENT = 1 << 4, SSL_SERVER = 1 << 5, SSL_DTLSCON = 1 << 6 }

Functions

- void ssl_shutdown (void *data)
- void * tlsv1_init (const char *cacert, const char *cert, const char *key, int verify)
- void * sslv2_init (const char *cacert, const char *cert, const char *key, int verify)
- void * sslv3_init (const char *cacert, const char *cert, const char *key, int verify)
- void * dtlsv1_init (const char *cacert, const char *cert, const char *key, int verify)
- void tlsaccept (struct fwsocket *sock, struct ssldata *orig)
- int socketread_d (struct fwsocket *sock, void *buf, int num, union sockstruct *addr)
- int socketread (struct fwsocket *sock, void *buf, int num)
- int socketwrite_d (struct fwsocket *sock, const void *buf, int num, union sockstruct *addr)
- int socketwrite (struct fwsocket *sock, const void *buf, int num)
- void sslstartup (void)
- void dtsl_serveropts (struct fwsocket *sock)
- struct fwsocket * dtls_listenssl (struct fwsocket *sock)
- void startsslclient (struct fwsocket *sock)
- void dtlstimeout (struct fwsocket *sock, struct timeval *timeleft, int defusec)
- void dtlshandltimeout (struct fwsocket *sock)

7.41.1 Macro Definition Documentation

7.41.1.1 #define COOKIE_SECRET_LENGTH 32

Definition at line 55 of file sslutil.c.

Referenced by sslstartup().

7.41.2 Enumeration Type Documentation

7.41.2.1 enum SSLFLAGS

Enumerator:

SSL_TLSV1

SSL_SSLV2

SSL_SSLV3

SSL_DTLSV1

SSL_CLIENT

SSL_SERVER

SSL_DTLSCON

Definition at line 36 of file sslutil.c.

```
            {
    SSL_TLSV1   = 1 << 0,
    SSL_SSLV2   = 1 << 1,
    SSL_SSLV3   = 1 << 2,
    SSL_DTLSV1     = 1 << 3,
    SSL_CLIENT     = 1 << 4,
    SSL_SERVER     = 1 << 5,
    SSL_DTLSCON   = 1 << 6
};
```

7.41.3 Function Documentation

7.41.3.1 struct fwsocket* dtls_listenssl (struct fwsocket * *sock*) [read]

Definition at line 530 of file sslutil.c.

References fwsocket::addr, ssldata::flags, make_socket(), objalloc(), objlock(), objunlock(), objunref(), fwsocket::proto, sockstruct::sa, fwsocket::sock, ssldata::ssl, fwsocket::ssl, SSL_DTLSCON, and fwsocket::type.

```
                                                                {
    struct ssldata *ssl = sock->ssl;
    struct ssldata *newssl;
    struct fwsocket *newsock;
    union sockstruct client;
#ifndef __WIN32__
    int on = 1;
#endif

    if (!(newssl = objalloc(sizeof(*newssl), free_ssldata))) {
        return NULL;
    }

    newssl->flags |= SSL_DTLSCON;

    dtlssetopts(newssl, ssl, sock);
    memset(&client, 0, sizeof(client));
    if (DTLSv1_listen(newssl->ssl, &client) <= 0) {
        objunref(newssl);
        return NULL;
    }

    objlock(sock);
```

```
    if (!(newsock = make_socket(sock->addr.sa.sa_family, sock
      ->type, sock->proto, newssl))) {
        objunlock(sock);
        objunref(newssl);
        return NULL;
    }
    objunlock(sock);
    memcpy(&newsock->addr, &client, sizeof(newsock->addr));
#ifndef __WIN32__
    setsockopt(newsock->sock, SOL_SOCKET, SO_REUSEADDR, &on, sizeof(on));
#ifdef SO_REUSEPORT
    setsockopt(newsock->sock, SOL_SOCKET, SO_REUSEPORT, &on, sizeof(on));
#endif
#endif
    objlock(sock);
    bind(newsock->sock, &sock->addr.sa, sizeof(sock->addr));
    objunlock(sock);
    connect(newsock->sock, &newsock->addr.sa, sizeof(newsock->addr
      ));

    dtlsaccept(newsock);

    return (newsock);
}
```

7.41.3.2 void dtlshandltimeout (struct fwsocket * *sock*)

Definition at line 630 of file sslutil.c.

References objlock(), objunlock(), ssldata::ssl, and fwsocket::ssl.

```
                                                    {
    if (!sock->ssl) {
        return;
    }

    objlock(sock->ssl);
    DTLSv1_handle_timeout(sock->ssl->ssl);
    objunlock(sock->ssl);
}
```

7.41.3.3 void dtlstimeout (struct fwsocket * *sock,* struct timeval * *timeleft,* int *defusec*)

Definition at line 617 of file sslutil.c.

References objlock(), objunlock(), ssldata::ssl, and fwsocket::ssl.

```
          {
    if (!sock || !sock->ssl || !sock->ssl->ssl) {
        return;
    }

    objlock(sock->ssl);
    if (!DTLSv1_get_timeout(sock->ssl->ssl, timeleft)) {
        timeleft->tv_sec = 0;
        timeleft->tv_usec = defusec;
    }
    objunlock(sock->ssl);
}
```

7.41.3.4 void* dtlsv1_init (const char * *cacert,* const char * *cert,* const char * *key,* int *verify*)

Definition at line 237 of file sslutil.c.

References ssldata::ctx, ssldata::ssl, and SSL_DTLSV1.

```
                {
    const SSL_METHOD *meth = DTLSv1_method();
    struct ssldata *ssl;
```

```
    ssl = sslinit(cacert, cert, key, verify, meth, SSL_DTLSV1);
    /* XXX BIO_CTRL_DGRAM_MTU_DISCOVER*/
    SSL_CTX_set_read_ahead(ssl->ctx, 1);

    return (ssl);
}
```

7.41.3.5 void dtls_serveropts (struct fwsocket * *sock*)

Definition at line 489 of file sslutil.c.

References ssldata::ctx, ssldata::flags, objlock(), objunlock(), ssldata::ssl, fwsocket::ssl, and SSL_SERVER.

Referenced by socketserver().

```
                                                    {
    struct ssldata *ssl = sock->ssl;

    if (!ssl) {
        return;
    }

    dtlssetopts(ssl, NULL, sock);

    objlock(ssl);
    SSL_CTX_set_cookie_generate_cb(ssl->ctx, generate_cookie);
    SSL_CTX_set_cookie_verify_cb(ssl->ctx, verify_cookie);
    SSL_CTX_set_session_cache_mode(ssl->ctx, SSL_SESS_CACHE_OFF);

    SSL_set_options(ssl->ssl, SSL_OP_COOKIE_EXCHANGE);
    ssl->flags |= SSL_SERVER;
    objunlock(ssl);
}
```

7.41.3.6 int socketread (struct fwsocket * *sock,* void * *buf,* int *num*)

Definition at line 362 of file sslutil.c.

References socketread_d().

```
                                                                {
    return (socketread_d(sock, buf, num, NULL));
}
```

7.41.3.7 int socketread_d (struct fwsocket * *sock,* void * *buf,* int *num,* union sockstruct * *addr*)

Definition at line 297 of file sslutil.c.

References fwsocket::flags, objlock(), objunlock(), objunref(), sockstruct::sa, fwsocket::sock, SOCK_FLAG_CLOSE, ssldata::ssl, fwsocket::ssl, and fwsocket::type.

Referenced by socketread().

```
                {
    struct ssldata *ssl = sock->ssl;
    socklen_t salen = sizeof(*addr);
    int ret, err, syserr;

    if (!ssl || !ssl->ssl) {
        objlock(sock);
        if (addr && (sock->type == SOCK_DGRAM)) {
            ret = recvfrom(sock->sock, buf, num, 0, &addr->sa, &salen);
        } else {
            ret = read(sock->sock, buf, num);
        }
        if (ret == 0) {
            sock->flags |= SOCK_FLAG_CLOSE;
        }
```

```
        objunlock(sock);
        return (ret);
    }

    objlock(ssl);
    /* ive been shutdown*/
    if (!ssl->ssl) {
        objunlock(ssl);
        return (-1);
    }
    ret = SSL_read(ssl->ssl, buf, num);
    err = SSL_get_error(ssl->ssl, ret);
    if (ret == 0) {
        sock->flags |= SOCK_FLAG_CLOSE;
    }
    objunlock(ssl);
    switch (err) {
        case SSL_ERROR_NONE:
            break;
        case SSL_ERROR_WANT_X509_LOOKUP:
            printf("Want X509\n");
            break;
        case SSL_ERROR_WANT_READ:
            printf("Want Read\n");
            break;
        case SSL_ERROR_WANT_WRITE:
            printf("Want write\n");
            break;
        case SSL_ERROR_ZERO_RETURN:
        case SSL_ERROR_SSL:
            objlock(sock);
            objunref(sock->ssl);
            sock->ssl = NULL;
            objunlock(sock);
            break;
        case SSL_ERROR_SYSCALL:
            syserr = ERR_get_error();
            if (syserr || (!syserr && (ret == -1))) {
                printf("R syscall %i %i\n", syserr, ret);
            }
            break;
        default
                :
            printf("other\n");
            break;
    }

    return (ret);
}
```

7.41.3.8 int socketwrite (struct fwsocket * *sock,* const void * *buf,* int *num*)

Definition at line 444 of file sslutil.c.

References socketwrite_d().

```
                                                                    {
    return (socketwrite_d(sock, buf, num, NULL));
}
```

7.41.3.9 int socketwrite_d (struct fwsocket * *sock,* const void * *buf,* int *num,* union sockstruct * *addr*)

Definition at line 366 of file sslutil.c.

References fwsocket::flags, objlock(), objunlock(), objunref(), sockstruct::sa, setflag, fwsocket::sock, SOCK_FLAG_CLOSE, ssldata::ssl, fwsocket::ssl, and fwsocket::type.

Referenced by socketwrite().

```
                        {
    struct ssldata *ssl = (sock) ? sock->ssl : NULL;
    int ret, err, syserr;

    if (!sock) {
        return (-1);
```

```
    }

#ifndef __WIN32__
    if (!ssl || !ssl->ssl) {
        objlock(sock);
        if (addr && (sock->type == SOCK_DGRAM)) {
            ret = sendto(sock->sock, buf, num, MSG_NOSIGNAL, &addr->sa,
      sizeof(*addr));
        } else {
            ret = send(sock->sock, buf, num, MSG_NOSIGNAL);
        }
        if (ret == -1) {
            switch(errno) {
                case EBADF:
                case EPIPE:
                case ENOTCONN:
                case ENOTSOCK:
                    sock->flags |= SOCK_FLAG_CLOSE;
                    break;
            }
        }
        objunlock(sock);
        return (ret);
    }
#endif

    objlock(ssl);
    if (SSL_state(ssl->ssl) != SSL_ST_OK) {
        objunlock(ssl);
        return (SSL_ERROR_SSL);
    }
    ret = SSL_write(ssl->ssl, buf, num);
    err = SSL_get_error(ssl->ssl, ret);
    objunlock(ssl);

    if (ret == -1) {
        setflag(sock, SOCK_FLAG_CLOSE);
    }

    switch(err) {
        case SSL_ERROR_NONE:
            break;
        case SSL_ERROR_WANT_READ:
            printf("Want Read\n");
            break;
        case SSL_ERROR_WANT_WRITE:
            printf("Want write\n");
            break;
        case SSL_ERROR_WANT_X509_LOOKUP:
            printf("Want X509\n");
            break;
        case SSL_ERROR_ZERO_RETURN:
        case SSL_ERROR_SSL:
            objlock(sock);
            objunref(sock->ssl);
            sock->ssl = NULL;
            objunlock(sock);
            break;
        case SSL_ERROR_SYSCALL:
            syserr = ERR_get_error();
            if (syserr || (!syserr && (ret == -1))) {
                printf("W syscall %i %i\n", syserr, ret);
            }
            break;
        default
                :
            printf("other\n");
            break;
    }

    return (ret);
}
```

7.41.3.10 void ssl_shutdown (void * *data*)

Definition at line 92 of file sslutil.c.

References objlock(), objunlock(), and ssldata::ssl.

```
                                    {
    struct ssldata *ssl = data;
    int err, ret;
```

```
    if (!ssl) {
        return;
    }

    objlock(ssl);
    if (ssl->ssl && ((ret = SSL_shutdown(ssl->ssl)) < 1)) {
        objunlock(ssl);
        if (ret == 0) {
            objlock(ssl);
            ret = SSL_shutdown(ssl->ssl);
        } else {
            objlock(ssl);
        }
        err = SSL_get_error(ssl->ssl, ret);
        switch(err) {
            case SSL_ERROR_WANT_READ:
                printf("SSL_shutdown wants read\n");
                break;
            case SSL_ERROR_WANT_WRITE:
                printf("SSL_shutdown wants write\n");
                break;
            case SSL_ERROR_SSL:
                /*ignore im going away now*/
            case SSL_ERROR_SYSCALL:
                /* ignore this as documented*/
            case SSL_ERROR_NONE:
                /* nothing to see here moving on*/
                break;
            default
                    :
                printf("SSL Shutdown unknown error %i\n", err);
                break;
        }
    }
    if (ssl->ssl) {
        SSL_free(ssl->ssl);
        ssl->ssl = NULL;
    }
    objunlock(ssl);
}
```

7.41.3.11 void sslstartup (void)

Definition at line 448 of file sslutil.c.

References COOKIE_SECRET_LENGTH, genrand(), and malloc.

Referenced by framework_init().

```
                              {
    SSL_library_init();
    SSL_load_error_strings();
    OpenSSL_add_ssl_algorithms();

    if ((cookie_secret = malloc(COOKIE_SECRET_LENGTH)
      )) {
        genrand(cookie_secret, COOKIE_SECRET_LENGTH)
      ;
    }
}
```

7.41.3.12 void* sslv2_init (const char * *cacert*, const char * *cert*, const char * *key*, int *verify*)

Definition at line 221 of file sslutil.c.

References SSL_SSLV2.

```
                  {
    const SSL_METHOD *meth = SSLv2_method();

    return (sslinit(cacert, cert, key, verify, meth, SSL_SSLV2));
}
```

7.41.3.13 void* sslv3_init (const char * *cacert*, const char * *cert*, const char * *key*, int *verify*)

Definition at line 228 of file sslutil.c.

References ssldata::ssl, and SSL_SSLV3.

```
                {
    const SSL_METHOD *meth = SSLv3_method();
    struct ssldata *ssl;

    ssl = sslinit(cacert, cert, key, verify, meth, SSL_SSLV3);

    return (ssl);
}
```

7.41.3.14 void startsslclient (struct fwsocket * *sock*)

Definition at line 602 of file sslutil.c.

References ssldata::flags, fwsocket::ssl, SSL_SERVER, and fwsocket::type.

Referenced by socketclient().

```
                                                    {
    if (!sock || !sock->ssl || (sock->ssl->flags & SSL_SERVER
      )) {
        return;
    }

    switch(sock->type) {
        case SOCK_DGRAM:
            dtlsconnect(sock);
            break;
        case SOCK_STREAM:
            sslsockstart(sock, NULL, 0);
            break;
    }
}
```

7.41.3.15 void tlsaccept (struct fwsocket * *sock*, struct ssldata * *orig*)

Definition at line 290 of file sslutil.c.

References objalloc(), and fwsocket::ssl.

```
                                                                    {
    if ((sock->ssl = objalloc(sizeof(*sock->ssl), free_ssldata)))
       {
        sslsockstart(sock, orig, 1);
    }

}
```

7.41.3.16 void* tlsv1_init (const char * *cacert*, const char * *cert*, const char * *key*, int *verify*)

Definition at line 214 of file sslutil.c.

References SSL_TLSV1.

```
                {
    const SSL_METHOD *meth = TLSv1_method();

    return (sslinit(cacert, cert, key, verify, meth, SSL_TLSV1));
}
```

7.42 src/thread.c File Reference

Functions for starting and managing threads.

```
#include <pthread.h>
#include <signal.h>
#include <unistd.h>
#include <stdint.h>
#include "include/dtsapp.h"
```

Data Structures

- struct thread_pvt

  thread struct used to create threads data needs to be first element
- struct threadcontainer

  Global threads data.

Macros

- #define SIGHUP 1

  Define SIGHUP as 1 if its not defined.
- #define THREAD_MAGIC 0xfeedf158

  32 bit magic value to help determine thread is ok

Enumerations

- enum threadopt { TL_THREAD_NONE = 1 << 0, TL_THREAD_RUN = 1 << 1, TL_THREAD_DONE = 1 << 2 }

  Thread status a thread can be disabled by unsetting TL_THREAD_RUN.

Functions

- int framework_threadok (void *data)

  let threads check there status by passing in a pointer to there data
- int startthreads (void)

  initialise the threadlist start manager thread
- void stopthreads (void)

  Stoping the manager thread will stop all other threads.
- struct thread_pvt * framework_mkthread (threadfunc func, threadcleanup cleanup, threadsighandler sig_handler, void *data)

  create a thread result must be unreferenced
- void jointhreads (void)

  Join the manager thread.

Variables

- struct threadcontainer * threads = NULL

  Thread control data.

7.42.1 Detailed Description

Functions for starting and managing threads. The thread interface consists of a management thread managing a hashed bucket list of threads running optional clean up when done.

Definition in file thread.c.

7.43 src/unixsock.c File Reference

Attach a thread to a unix socket calling a callback on connect.

```
#include <sys/socket.h>
#include <sys/stat.h>
#include <linux/un.h>
#include <linux/limits.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include "include/dtsapp.h"
```

Data Structures

- struct framework_sockthread

  Unix socket data structure.

Functions

- void framework_unixsocket (char *sock, int protocol, int mask, threadfunc connectfunc, threadcleanup cleanup)

  Create and run UNIX socket thread.

7.43.1 Detailed Description

Attach a thread to a unix socket calling a callback on connect. A thread is started on the sockect and will start a new client thread on each connection with the socket as the data

Definition in file unixsock.c.

7.44 src/util.c File Reference

Utilities commonly used.

```
#include <openssl/bio.h>
#include <openssl/buffer.h>
#include <openssl/evp.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>
#include <openssl/rand.h>
#include <openssl/md5.h>
#include <openssl/sha.h>
#include <ctype.h>
#include <stdint.h>
#include <stdio.h>
#include <sys/time.h>
#include "include/dtsapp.h"
```

Functions

- void seedrand (void)

  Seed openssl random number generator.
- int genrand (void *buf, int len)

  Generate random sequence.
- void sha512sum2 (unsigned char *buff, const void *data, unsigned long len, const void *data2, unsigned long len2)

  Calculate the SHA2-512 hash accross 2 data chunks.
- void sha512sum (unsigned char *buff, const void *data, unsigned long len)

  Calculate the SHA2-512 hash.
- void sha256sum2 (unsigned char *buff, const void *data, unsigned long len, const void *data2, unsigned long len2)

  Calculate the SHA2-256 hash accross 2 data chunks.
- void sha256sum (unsigned char *buff, const void *data, unsigned long len)

  Calculate the SHA2-256 hash.
- void sha1sum2 (unsigned char *buff, const void *data, unsigned long len, const void *data2, unsigned long len2)

  Calculate the SHA1 hash accross 2 data chunks.
- void sha1sum (unsigned char *buff, const void *data, unsigned long len)

  Calculate the SHA1 hash.
- void md5sum2 (unsigned char *buff, const void *data, unsigned long len, const void *data2, unsigned long len2)

  Calculate the MD5 hash accross 2 data chunks.
- void md5sum (unsigned char *buff, const void *data, unsigned long len)

  Calculate the MD5 hash.
- int md5cmp (unsigned char *digest1, unsigned char *digest2)

  Compare two md5 hashes.
- int sha1cmp (unsigned char *digest1, unsigned char *digest2)

  Compare two SHA1 hashes.
- int sha256cmp (unsigned char *digest1, unsigned char *digest2)

  Compare two SHA2-256 hashes.
- int sha512cmp (unsigned char *digest1, unsigned char *digest2)

  Compare two SHA2-512 hashes.
- void md5hmac (unsigned char *buff, const void *data, unsigned long len, const void *key, unsigned long klen)

  Hash Message Authentication Codes (HMAC) MD5.

- void sha1hmac (unsigned char *buff, const void *data, unsigned long len, const void *key, unsigned long klen)

  Hash Message Authentication Codes (HMAC) SHA1.
- void sha256hmac (unsigned char *buff, const void *data, unsigned long len, const void *key, unsigned long klen)

  Hash Message Authentication Codes (HMAC) SHA2-256.
- void sha512hmac (unsigned char *buff, const void *data, unsigned long len, const void *key, unsigned long klen)

  Hash Message Authentication Codes (HMAC) SHA2-512.
- int strlenzero (const char *str)

  Check if a string is zero length.
- char * ltrim (char *str)

  Trim white space at the begining of a string.
- char * rtrim (const char *str)

  Trim white space at the end of a string.
- char * trim (const char *str)

  Trim whitesapce from the beggining and end of a string.
- uint64_t tvtontp64 (struct timeval *tv)

  Convert a timeval struct to 64bit NTP time.
- uint16_t checksum (const void *data, int len)

  Obtain the checksum for a buffer.
- uint16_t checksum_add (const uint16_t checksum, const void *data, int len)

  Obtain the checksum for a buffer adding a checksum.
- uint16_t verifysum (const void *data, int len, const uint16_t check)

  Verify a checksum.
- void touch (const char *filename, uid_t user, gid_t group)

  Create a file and set user and group.
- char * b64enc_buf (const char *message, uint32_t len, int nonl)

  Base 64 encode a buffer.
- char * b64enc (const char *message, int nonl)

  Base 64 encode a string.

7.44.1 Detailed Description

Utilities commonly used.

`@n @verbatim`

- Acknowledgments [MD5 HMAC http://www.ietf.org/rfc/rfc2104.txt]
- Pau-Chen Cheng, Jeff Kraemer, and Michael Oehler, have provided
- useful comments on early drafts, and ran the first interoperability
- tests of this specification. Jeff and Pau-Chen kindly provided the
- sample code and test vectors that appear in the appendix. Burt
- Kaliski, Bart Preneel, Matt Robshaw, Adi Shamir, and Paul van
- Oorschot have provided useful comments and suggestions during the
- investigation of the HMAC construction.
-

Definition in file util.c.

7.45 src/zlib.c File Reference

Simplified interface to Compress/Uncompress/Test a buffer.

```
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <zlib.h>
#include "include/dtsapp.h"
```

Functions

- struct zobj * zcompress (uint8_t *buff, uint16_t len, uint8_t level)

  Allocate a buffer and return it with compressed data.
- void zuncompress (struct zobj *buff, uint8_t *obuff)

  Uncompress zobj buffer to buffer.
- int is_gzip (uint8_t *buf, int buf_size)

  check a buffer if it contains gzip magic
- uint8_t * gzinflatebuf (uint8_t *buf_in, int buf_size, uint32_t *len)

  Ungzip a buffer.

7.45.1 Detailed Description

Simplified interface to Compress/Uncompress/Test a buffer.

Definition in file zlib.c.