GAA-API Reference Manual

version0.3

Generated by Doxygen 1.2.1

Wed Jun 27 16:14:57 2001

Contents

Chapter 1

GAA-API Module Index

1.1 GAA-API Modules

Here is a list of all modules:

Chapter 2

GAA-API Overview

2.1 GAA-API Overview

The Generic Authorization and Access Control Application Programming Interface (GAA-API) provides access control services to calling applications. It allows applications to make access control decisions based on access control policies associated with a targeted resource.

The GAA-API supports:

- various security mechanisms based on public or secret key cryptosystems
- different authorization models
- heterogeneous security policies
- various access rights

2.2 GAA-API Library

libgaa_core.so
- Several routines for returning string representation of condition status, credential type, and the right types are implemented.
- Also, various routines have been implemented to check whether a specific callback has been set for a GAA pointer

libgaa_plugin.so
- Various plugin routines to further define GAA behavior (e.g. find the subset of a policy that matches a request right, and add that subset to an output policy) have been implemented.

libgaa_simple.so
- A simple routines to check identity/group conditions, to pull assertion credentials, to evaluate an assertion credentia, and etc. are implemented.

libgaa_pthread.so
- Thread-specific routines (e.g. create, destroy, lock and unlock a pthread) are implemented.

libgaa_util.so
- An implementation of string token operation (e.g. gaautil_gettok - this is conceptually similar to ”strtok”)

libgaa_supp.so
- A portable implementation of snprintf (a routine that converts numeric and string arguments to formatted strings).

- Some newer operating systems implements snprintf in their C library, but many do not provide adequate version.

Chapter 3

Installation

3.1 Where to download

The GAA-API standalone version is under the following URL:

URL http://www.isi.edu/gost/info/gaaapi/

3.2 How to compile

NOTE 1 This version was tested on Solaris 2.5.1, Solaris 2.7, and Linux 2.4.

NOTE 2 GAA API requires GNU Libtool (http://www.gnu.org/software/libtool/). Libtool comes with standard release of Linux, but not with Solaris.

NOTE 3 In Solaris, make sure LD_LIBRARY_PATH is set to point the directory that has libtool shared library (libltdl.so) e.g., LD_LIBRARY_PATH=/nfs/tonga/zmbx/local/lib

STEP 1 Edit $SRCDIR in stand-alone/Makefile so that it points the source file directory.

STEP 2 Run ./pre-compile script to create a proper Makefile.common file.

STEP 3 Edit $LIBTOOLDIR in stand-alone/Makefile.common so that it points the Libtool directory.

STEP 4 Run make to compile.

Comment 1 Library files will be created under lib/

Comment 2 If your host is not in isi.edu domain, you also need to modify test/config/gaa.sparc—linux.cf file, so that the directories point the correct ones.

Comment 3 e.g., The following three lines in the gaa.sparc.cf file should be properly edited:

- libdir /nfs/ruby/gost3/zmbx/src/gaa/stand-alone/lib/
- params /nfs/ruby/gost3/zmbx/src/gaa/stand-alone/test/groups
- params /nfs/ruby/gost3/zmbx/src/gaa/stand-alone/test/eacls

3.3 How to test

The sample program is located under the directory called test/

- In linux, run ./linux-gaatest.
- In Solaris, run ./sparc-gaatest.

Several commands can be specified on the test program:

- `print sc` print GAA security context
- `getpolicy <file>` Read policy information from the EACL contained in ¡policydir¿/¡file¿ (where ¡policydir¿ is the directory specified in the config file).
- `print policy` print the current poicy
- `request <auth> <val>` request the right described by the specified authority and value (e.g. ”request file read”).
- `assert <name>` if the assertion mechanism is specified in the gss-server configuration file, add an assertion credential (with the specified name) to the security context.
- `CLEAR SC` clear the GAA security context.

3.4 Sample program

The following sample program is from Laura Pearlman’s GAA-API source code.

The entire source files, header files, sample policy files, and configuration files are available at the download site (see section 3.1).

To run the sample program, please refer to the section 3.3.

3.4.1 gaatest.c program

```
#include "gaa.h"
#include "gaa_simple.h"
#include "gaa_debug.h"
#include "gaa_utils.h"
#include <string.h>
#include <strings.h>


#define USAGE "Usage: %s cffile\n"

char *users = 0;

main(int argc, char **argv)
```

```
{
    gaa_status status;
    gaa_sc_ptr sc = 0;
    gaa_ptr gaa = 0;
    gaa_policy *policy = 0;
    char *ifname = 0;
    FILE *reqfile = stdin;
    gaa_list_ptr rlist;
    gaa_policy_right *pright;
    gaa_list_entry_ptr ent;
    char *object = 0;
    char buf[1024];
    char rbuf[8192];
    char *repl;
    char *what;
    char *cfname;

    switch(argc) {
    case 2:
cfname = argv[1];
break;
    default:
fprintf(stderr, USAGE, argv[0]);
exit(1);
    }

    if ((status = gaa_initialize(&gaa, (void *)cfname)) != GAA_S_SUCCESS) {
fprintf(stderr, "init_gaa failed: %s: %s\n",
gaa_x_majstat_str(status), gaa_get_err());
exit(1);
    }

    if ((status = gaa_new_sc(&sc)) != GAA_S_SUCCESS) {
fprintf(stderr, "gaa_new_sc failed: %s: %s\n",
gaa_x_majstat_str(status), gaa_get_err());
exit(1);
    }

    printf("> ");
    while (fgets(buf, sizeof(buf), stdin)) {
repl = process_msg(gaa, &sc, buf, rbuf, sizeof(rbuf), &users, &policy);
if (repl == 0)
    printf("(null reply)");
else
    printf("%s", repl);
printf("> ");
    }
    exit(0);
}
```

3.4.2 gaa_utils.c program

```
#include <string.h>
#include "gaa.h"
#include "gaa_core.h"
#include "gaa_utils.h"
#include "gaa_debug.h"

char *
process_msg(gaa_ptr gaa, gaa_sc_ptr *sc, char *inbuf, char *outbuf, int outbsize, char *
{
    char *what;

    *outbuf = '\0';
    if (what = strtok(inbuf, " \t\n")) {
if (strcasecmp(what, "assert") == 0)
    process_assert(gaa, *sc, users, outbuf, outbsize);
else if (strcasecmp(what, "getpolicy") == 0)
    process_getpolicy(gaa, policy, outbuf, outbsize);
else if (strcasecmp(what, "request") == 0) {
    if (inbuf = strtok(0, "\n"))
process_request(gaa, *sc, *policy, inbuf, outbuf, outbsize);
}
else if (strcasecmp(what, "print") == 0)
    process_print(gaa, *sc, policy, outbuf, outbsize);
else if (strcasecmp(what, "inquire") == 0)
    process_inquire(gaa, *sc, policy, outbuf, outbsize);
else if (strcasecmp(what, "clear") == 0)
    process_clear(sc);
else if (strcasecmp(what, "pull") == 0)
    process_pull(gaa, *sc, outbuf, outbsize);
else
    snprintf(outbuf, outbsize, "huh?\n");
    }
    return(outbuf);
}

gaa_status
process_assert(gaa_ptr gaa, gaa_sc_ptr sc, char **users, char *outbuf, int outbsize)
{
    char *name;
    gaa_status status = GAA_S_SUCCESS;

    if (name = strtok(0, " \t\n")) {
*users = name;
if ((status = gaa_pull_creds(gaa, sc, GAA_ANY, 0)) != GAA_S_SUCCESS)
    snprintf(outbuf, outbsize, "pull_creds failed: %s (%s)\n",
     gaacore_majstat_str(status),
     gaa_get_err());
    }
    return(status);
```

```
}

gaa_status
process_pull(gaa_ptr gaa, gaa_sc_ptr sc, char *out, int osize)
{
    char *mech = 0;
    char *tstr = 0;
    gaa_cred_type type;
    char *s;
    int len;
    gaa_status status;

    struct slist {
char *name;
gaa_cred_type val;
    };

    struct slist types[] = {
{"identity", GAA_IDENTITY},
{"group", GAA_GROUP_MEMB},
{"group-non", GAA_GROUP_NON_MEMB},
{"authorized", GAA_AUTHORIZED},
{"attributes", GAA_ATTRIBUTES},
{"uneval", GAA_UNEVAL},
{"any", GAA_ANY},
{0, GAA_UNEVAL}
    };
    struct slist *sl;

    if (mech = strtok(0, " \t\n"))
tstr = strtok(0, "\t\n");
    if (mech && (strcmp(mech, "0") == 0))
mech = 0;
    if (tstr) {
for (sl = types; sl->name; sl++)
    if (strcasecmp(sl->name, tstr) == 0) {
type = sl->val;
break;
    }
if (sl->name == 0) {
    s = out;
    snprintf(s, osize, "Invalid cred type; please choose one of");
    len = strlen(s);
    s += len;
    osize -= len;
    for (sl = types; sl->name; sl++) {
snprintf(s, osize, " %s", sl->name);
len = strlen(s);
s += len;
osize -= len;
    }
    snprintf(s, osize, "\n");
```

```
    return(GAA_STATUS(GAA_S_INVALID_ARG, 0));
}
    } else
type = GAA_ANY;

    if ((status = gaa_pull_creds(gaa, sc, type, mech)) != GAA_S_SUCCESS)
snprintf(out, osize, "pull_creds failed: %s (%s)\n",
gaacore_majstat_str(status), gaa_get_err());
    return(status);
}

gaa_status
process_getpolicy(gaa_ptr gaa, gaa_policy_ptr *policy, char *outbuf, int outbsize)
{
    char *object;
    gaa_status status;

    if ((object = strtok(0, "\t\n")) == 0) {
fprintf(stderr, "no object specified\n");
return(GAA_STATUS(GAA_S_NO_MATCHING_ENTRIES, 0));
    }
    gaa_clear_policy(*policy);
    if ((status = gaa_get_object_policy_info(object, gaa, policy)) != GAA_S_SUCCESS)
snprintf(outbuf, outbsize, "gaa_get_object_policy_info failed: %s (%s)\n",
gaacore_majstat_str(status), gaa_get_err());
    return(status);
}

gaa_status
process_request(gaa_ptr gaa, gaa_sc_ptr sc, gaa_policy_ptr policy, char *inbuf, char *ou
{
    char *auth;
    char *val;
    gaa_status status;
    gaa_list_ptr list = 0;
    gaa_request_right *right;
    gaa_answer_ptr answer;
    char *statstr;
    char *str;
    char *s1;
    int len;

    str = inbuf;
    while (str && *str) {
if (s1 = strpbrk(str, ";\n"))
    *s1++ = '\0';
if ((auth = strtok(str, " \t\n")) == 0)
    continue;
if (strcasecmp(auth, "end") == 0)
    break;
if ((val = strtok(0, " \t\n")) == 0) {
    snprintf(outbuf, outbsize, "No value specified\n");
```

```
    break;
}
if (list == 0)
    list = gaa_new_req_rightlist();
if ((status = gaa_new_request_right(gaa, &right, auth, val)) != GAA_S_SUCCESS) {
    snprintf(outbuf, outbsize, "gaa_new_request_right failed: %s (%s)\n",
    gaacore_majstat_str(status), gaa_get_err());
    return(status);
}
if ((status = gaa_add_request_right(list, right)) != GAA_S_SUCCESS) {
    snprintf(outbuf, outbsize, "gaa_add_request_right failed: %s (%s)\n",
    gaacore_majstat_str(status), gaa_get_err());
    return(status);
}
str = s1;
    }
    if ((status = gaa_new_answer(&answer)) != GAA_S_SUCCESS) {
snprintf(outbuf, outbsize, "gaa_new_answer failed: %s (%s)\n",
gaacore_majstat_str(status), gaa_get_err());
return(status);
    }
    status = gaa_check_authorization(gaa, sc, policy, list, answer);
    if (status == GAA_C_YES)
statstr = "GAA_C_YES";
    else
statstr = gaacore_majstat_str(status);
    switch(status) {
    case GAA_C_YES:
    case GAA_C_NO:
    case GAA_C_MAYBE:
snprintf(outbuf, outbsize, "%s -- Detailed answer:\n", statstr);
len = strlen(outbuf);
str = outbuf + len;
outbsize -= len;
gaadebug_answer_string(gaa, str, outbsize, answer);
break;
    default:
snprintf(outbuf, outbsize,
 "gaa_check_authorization returned %s: %s\n", statstr,
       gaa_get_err());
    }
    return(status);
}

void
process_print(gaa_ptr gaa, gaa_sc_ptr sc, gaa_policy_ptr *policy, char *outbuf, int outl
{
    char *what;

    if ((what = strtok(0, " \t\n")) == 0)
snprintf(outbuf, outbsize, "Print what?\n");
    else if (strcasecmp(what, "sc") == 0)
```

```
gaadebug_sc_string(gaa, sc, outbuf, outbsize);
    else if (strcasecmp(what, "policy") == 0) {
if (policy)
    gaadebug_policy_string(gaa, outbuf, outbsize, *policy);
else
    snprintf(outbuf, outbsize, "(null policy)\n");
    } else
snprintf(outbuf, outbsize, "Print what?\n");
}

gaa_status
process_inquire(gaa_ptr gaa, gaa_sc_ptr sc, gaa_policy_ptr *policy, char *outbuf, int ou
{
    gaa_status status;
    gaa_list_ptr orights;
    gaa_list_entry_ptr ent;
    gaa_policy_right *pright;
    char *str;
    int len;

    if (policy == 0) {
snprintf(outbuf, outbsize, "(null policy)\n");
return(GAA_S_SUCCESS);
    }
    if ((status = gaa_inquire_policy_info(gaa, sc, *policy, &orights)) != GAA_S_SUCCESS)
snprintf(outbuf, outbsize, "gaa_inquire_policy_info failed: %s: %s\n",
gaacore_majstat_str(status), gaa_get_err());
return(status);
    }

    str = outbuf;
    snprintf(outbuf, outbsize, "policy info:\n");
    len = strlen(str);
    str += len;
    if ((outbsize -= len) < 2)
return(GAA_STATUS(GAA_S_INTERNAL_ERR, 0));
    for (ent = gaa_list_first(orights); ent; ent = gaa_list_next(ent)) {
pright = (gaa_policy_right *)gaa_list_entry_value(ent);
str = gaadebug_policy_right_string(gaa, str, outbsize, pright);
len = strlen(str);
str += len;
if ((outbsize -= len) < 2)
    return(GAA_STATUS(GAA_S_INTERNAL_ERR, 0));
    }
    return(status);
}

gaa_status
process_clear(gaa_sc_ptr *sc)
{
    gaa_free_sc(*sc);
    return(gaa_new_sc(sc));
```

```
}

init_sc(gaa_ptr gaa, gaa_sc_ptr *sc, void *context)
{
    gaa_cred *cred;
    /*
     * context is gss_ctx_id_t -- but I don't want to have to include
     * all the gss stuff here.
     */

    if (gaa_new_sc(sc) != GAA_S_SUCCESS)
return(-1);
    if (gaa_new_cred(gaa, *sc, &cred, "gss", context,
     GAA_IDENTITY, 1, 0) != GAA_S_SUCCESS)
return(-1);

    if (gaa_add_cred(gaa, *sc, cred) != GAA_S_SUCCESS)
return(-1);
    return(0);
}
```

3.4.3 Configuation file

```
libdir /nfs/ruby/gost3/zmbx/src/gaa/stand-alone/lib/
mechinfo assertion {
   cred_pull libgaa_simple.so gaasimple_assert_cred_pull
   cred_eval libgaa_simple.so gaasimple_assert_cred_eval
   cred_verify libgaa_simple.so gaasimple_assert_cred_verify
   sym_params SELF users
}
cond_eval cond_access_id assertion {
   cond_eval libgaa_simple.so gaasimple_check_id_cond
   idcred yes
   sym_params libgaa_simple.so gaa_identity
}

cond_eval cond_access_id x509 {
   cond_eval libgaa_simple.so gaasimple_check_id_cond
   idcred yes
   sym_params libgaa_simple.so gaa_identity
}

cond_eval cond_access_id_anyone DEFAULT {
   cond_eval libgaa_simple.so gaasimple_check_id_cond
   idcred yes
   sym_params libgaa_simple.so gaa_identity
}

cond_eval cond_access_group simple {
```

```
   cond_eval libgaa_simple.so gaasimple_check_group_cond
   idcred yes
   params /nfs/ruby/gost3/zmbx/src/gaa/stand-alone/test/groups
}

getpolicy {
   getpolicy libgaa_simple.so gaasimple_read_eacl
   params /nfs/ruby/gost3/zmbx/src/gaa/stand-alone/test/eacls
   freeparam /lib/libc.so.6 free
}
```

3.4.4 Policy file

```
pos_access_right file execute
cond_access_id assertion mei

neg_access_right file read
cond_group_id unix griduser

pos_access_right file write
cond_group_id unix griduser

neg_access_right file read
cond_access_id assertion laura

pos_access_right file write
cond_access_id assertion laura

neg_access_right file write
cond_access_id assertion mei

pos_access_right file read
cond_access_id assertion mei
```

Chapter 4

GAA-API Module Documentation

4.1 ”gaa”

Functions

- gaa_status gaa_new_condition (gaa_condition ** cond, gaa_string_data type, gaa_string_data authority, gaa_string_data value)

  gaa_new_condition().

- void gaa_free_condition (gaa_condition *cond)

  gaa_free_condition().

- gaa_status gaa_new_request_right (gaa_ptr gaa, gaa_request_right **right, gaa_string_data authority, gaa_string_data val)

  gaa_new_request_right().

- gaa_status gaa_new_request_right_rawval (gaa_ptr gaa, gaa_request_right **right, gaa_string_data authority, void * value)

  gaa_new_request_right_rawval().

- void gaa_free_request_right (gaa_request_right *right)

  gaa_free_request_right().

- gaa_status gaa_add_condition (gaa_policy_right * right, gaa_condition * condition)

  gaa_add_condition().

- gaa_status gaa_new_gaa (gaa_ptr *gaa)

  gaa_new_gaa().

- gaa_status gaa_new_sc (gaa_sc_ptr *sc)

  gaa_new_sc().

- gaa_status gaa_new_sec_attrb (gaa_sec_attrb ** a, gaa_cred_type type, gaa_string_data authority, gaa_string_data value)

  gaa_new_sec_attrb().

- void gaa_free_sec_attrb (gaa_sec_attrb *a)

  gaa_free_sec_attrb().

- gaa_status gaa_new_identity_info (gaa_ptr gaa, gaa_identity_info **info)

  gaa_new_identity_info().

- gaa_status gaa_new_cred (gaa_ptr gaa, gaa_sc_ptr sc, gaa_cred ** cred, gaa_string_data mech_type, void * mech_spec_cred, gaa_cred_type cred_type, int evaluate, gaa_status * estat)

  gaa_new_cred().

- void gaa_free_identity_info (gaa_identity_info *info)

  gaa_free_identity_info().

- gaa_status gaa_new_authr_info (gaa_ptr gaa, gaa_authr_info **info, void * objects, gaa_freefunc free_objects)

  gaa_new_authr_info().

- void gaa_free_authr_info (gaa_authr_info *info)

  gaa_free_authr_info().

- gaa_status gaa_add_cred (gaa_ptr gaa, gaaint_sc *sc, gaa_cred * cred)

  gaa_add_cred().

- gaa_status gaa_set_getpolicy_callback (gaa_ptr gaa, gaa_getpolicy_func func, void * param, gaa_freefunc freefunc)

  gaa_set_getpolicy_callback().

- gaa_status gaa_set_matchrights_callback (gaa_ptr gaa, gaa_matchrights_func func, void * param, gaa_freefunc freefunc)

  gaa_set_matchrights_callback().

- void gaa_free_sc (gaa_sc_ptr sc)

  gaa_free_sc().

- void gaa_free_gaa (gaa_ptr gaa)

  gaa_free_gaa().

- gaa_status gaa_add_mech_info (gaa_ptr gaa, gaa_string_data mech_type, gaa_cred_pull_func cred_pull, gaa_cred_eval_func cred_eval, gaa_cred_verify_func cred_verify, gaa_freefunc cred_free, void * param, gaa_freefunc freeparams)

  gaa_add_mech_info().

- gaa_status gaa_get_object_policy_info (gaa_string_data object, gaa_ptr gaa, gaa_policy_ptr * policy)

  gaa_get_object_policy_info().

- gaa_status gaa_pull_creds (gaa_ptr gaa, gaa_sc_ptr sc, gaa_cred_type which, gaa_string_data mech_type)

gaa_pull_creds().

- gaa_status gaa_new_cond_eval_callback (gaa_cond_eval_callback_ptr *cb, gaa_cond_eval_func func, void * params, gaa_freefunc freefunc)

  gaa_new_cond_eval_callback().

- void gaa_free_cond_eval_callback (gaa_cond_eval_callback_ptr cb)

  gaa_free_cond_eval_callback().

- gaa_status gaa_add_cond_eval_callback (gaa_ptr gaa, gaa_cond_eval_callback_ptr cb, gaa_string_data type, gaa_string_data authority, int is_idcred)

  gaa_add_cond_eval_callback().

- gaa_status gaa_getcreds (gaa_ptr gaa, gaa_sc_ptr sc, gaa_list_ptr * credlist, gaa_cred_type which)

  gaa_getcreds().

- gaa_list_ptr gaa_new_req_rightlist ()

  gaa_new_req_rightlist().

- gaa_status gaa_add_request_right (gaa_list_ptr rightlist, gaa_request_right *right)

  gaa_add_request_right().

- gaa_status gaa_add_option (gaa_request_right * right, gaa_string_data type, gaa_string_data authority, void * value, gaa_freefunc freeval)

  gaa_add_option().

- void gaa_free_cred (gaa_cred *cred)

  gaa_free_cred().

- gaa_status gaa_new_attribute_info (gaa_ptr gaa, gaa_attribute_info **info, gaa_string_data type, gaa_string_data authority, gaa_string_data value)

  gaa_new_attribute_info().

- void gaa_free_attribute_info (gaa_attribute_info *info)

  gaa_free_attribute_info.

- gaa_status gaa_add_cred_condition (gaa_cred * cred, gaa_condition * cond)

  gaa_add_cred_condition().

- gaa_status gaa_add_authr_right (gaa_cred * cred, gaa_policy_right * right)

  gaa_add_authr_right().

- char* gaa_request_rightval_string (gaa_ptr gaa, char * authority, void * val, char * buf, int bsize)

  gaa_request_rightval_string().

- char* gaa_policy_rightval_string (gaa_ptr gaa, char *authority, void *val, char *buf, int bsize)

  gaa_policy_rightval_string().

- gaa_status gaa_verify_cred (gaa_cred *cred)

  gaa_verify_cred().

- gaa_status gaa_add_authinfo (gaa_ptr gaa, char * authority, gaa_valinfo_ptr pvinfo, gaa_valinfo_ptr rvinfo, gaa_valmatch_func match, void * params, gaa_freefunc freeparams)

  gaa_add_authinfo().

- gaa_status gaa_new_valinfo (gaa_valinfo_ptr * valinfo, gaa_copyval_func copyval, gaa_string2val_-func newval, gaa_freefunc freeval, gaa_val2string_func val2str)

  gaa_new_valinfo().

- void gaa_free_valinfo (gaa_valinfo_ptr valinfo)

  gaa_free_valinfo.

- gaa_list_entry_ptr gaa_list_first (gaa_list_ptr list)

  gaa_list_first().

- gaa_list_entry_ptr gaa_list_next (gaa_list_entry_ptr entry)

  gaa_list_next().

- void* gaa_list_entry_value (gaa_list_entry_ptr entry)

  gaa_list_entry_value().

- void gaa_list_free (gaa_list_ptr list)

  gaa_list_free().

- gaa_status gaa_check_authorization (gaa_ptr gaa, gaa_sc_ptr sc, gaa_policy_ptr policy, gaa_list_ptr req_rights, gaa_answer_ptr answer)

  gaa_check_authorization().

- gaa_status gaa_inquire_policy_info (gaa_ptr gaa, gaa_sc_ptr sc, gaa_policy_ptr policy, gaa_list_ptr * out_rights)

  gaa_inquire_policy_info().

- gaa_status gaa_match_rights (gaa_ptr gaa, gaa_request_right * rright, gaa_policy_right * pright, int * match)

  gaa_match_rights().

- gaa_status gaa_check_condition (gaa_ptr gaa, gaa_sc_ptr sc, gaa_condition * cond, gaa_time_period * vtp, int * ynm, gaa_list_ptr options)

  gaa_check_condition().

- gaa_status gaa_new_answer (gaa_answer **answer)

  gaa_new_answer().

- void gaa_free_answer (gaa_answer *answer)

  gaa_free_answer().

- gaa_status gaa_new_policy (gaa_policy **policy)

  gaa_new_policy().

- gaa_status gaa_init_policy (gaa_policy *policy)

gaa_init_policy().

- void gaa_clear_policy (gaa_policy *policy)

  gaa_clear_policy().

- void gaa_free_policy (gaa_policy *policy)

  gaa_free_policy().

- gaa_status gaa_add_policy_entry (gaa_policy * policy, gaa_policy_right * right, int priority, int num)

  gaa_add_policy_entry().

- void gaa_free_policy_entry (gaa_policy_entry *ent)

  gaa_free_policy_entry().

- gaa_status gaa_new_policy_right (gaa_ptr gaa, gaa_policy_right **right, gaa_right_type type, gaa_string_data authority, gaa_string_data val)

  gaa_new_policy_right().

- gaa_status gaa_new_policy_right_rawval (gaa_ptr gaa, gaa_policy_right ** right, gaa_right_type type, gaa_string_data authority, void * val)

  gaa_new_policy_right_rawval().

- void gaa_free_policy_right (gaa_policy_right *right)

  gaa_free_policy_right().

- char* gaa_get_err ()

  gaa_get_err().

- gaa_status gaa_set_callback_err (char *s)

  gaa_get_err().

- char* gaa_get_callback_err ()

  gaa_get_err().

- gaa_status gaa_add_authinfo (gaa_ptr gaa, char *authority, gaa_valinfo_ptr pvinfo, gaa_valinfo_ptr rvinfo, gaa_valmatch_func match, void *params, gaa_freefunc freeparams)

  gaa_add_authinfo().

- gaa_status gaa_new_valinfo (gaa_valinfo_ptr *valinfo, gaa_copyval_func copyval, gaa_string2val_func newval, gaa_freefunc freeval, gaa_val2string_func val2str)

  gaa_new_valinfo().

- gaa_status gaa_inquire_policy_info (gaa_ptr gaa, gaa_sc_ptr sc, gaa_policy_ptr policy, gaa_list_ptr *out_rights)

  gaa_inquire_policy_info().

- gaa_status gaa_match_rights (gaa_ptr gaa, gaa_request_right *rright, gaa_policy_right *pright, int *match)

  gaa_match_rights().

- gaa_status gaa_check_condition (gaa_ptr gaa, gaa_sc_ptr sc, gaa_condition *cond, gaa_time_period *vtp, int *ynm, gaa_list_ptr options)

  gaa_check_condition().

- gaa_status gaa_add_policy_entry (gaa_policy *policy, gaa_policy_right *right, int priority, int num)

  gaa_add_policy_entry().

- gaa_status gaa_new_policy_right_rawval (gaa_ptr gaa, gaa_policy_right **right, gaa_right_type type, gaa_string_data authority, void *val)

  gaa_new_policy_right_rawval().

4.1.1 Function Documentation

4.1.1.1 gaa_status gaa_add_authinfo (gaa_ptr *gaa*, char * *authority*, gaa_valinfo_ptr *pvinfo*, gaa_valinfo_ptr *rvinfo*, gaa_valmatch_func *match*, void * *params*, gaa_freefunc *freeparams*)

gaa_add_authinfo().

Add an authinfo callback. This callback will be used to interpret and compare policy right values.

Parameters:

gaa input/output gaa pointer

authority optional input authority that this callback applies to. If authority is null, this is considered the default authinfo callback for any authority that does not have a specific authinfo callback.

pvinfo input valinfo callback (see gaa_new_valinfo()) to be used for policy rights with this authority.

rvinfo input valinfo callback (see gaa_new_valinfo()) to be used for request rights with this authority.

match input callback function that takes a policy right and a request right, and determines whether the values match.

params optional input callback parameters passed to pvinfo->copyval, rvinfo->copyval, pvinfo->newval, rvinfo->newval, pvinfo->val2str, rvinfo->val2str, and match whenever they're called.

freeparams optional input function to free params when the gaa structure is freed.

4.1.1.2 gaa_status gaa_add_authinfo (gaa_ptr *gaa*, char * *authority*, gaa_valinfo_ptr *pvinfo*, gaa_valinfo_ptr *rvinfo*, gaa_valmatch_func *match*, void * *params*, gaa_freefunc *freeparams*)

gaa_add_authinfo().

Add an authinfo callback. This callback will be used to interpret and compare policy right values.

Parameters:

gaa input/output gaa pointer

authority optional input authority that this callback applies to. If authority is null, this is considered the default authinfo callback for any authority that does not have a specific authinfo callback.

pvinfo input valinfo callback (see gaa_new_valinfo()) to be used for policy rights with this authority.

rvinfo input valinfo callback (see gaa_new_valinfo()) to be used for request rights with this authority.

match input callback function that takes a policy right and a request right, and determines whether the values match.

params optional input callback parameters passed to pvinfo->copyval, rvinfo->copyval, pvinfo->newval, rvinfo->newval, pvinfo->val2str, rvinfo->val2str, and match whenever they're called.

freeparams optional input function to free params when the gaa structure is freed.

4.1.1.3 gaa_status gaa_add_authr_right (gaa_cred * *cred*, gaa_policy_right * *right*)

gaa_add_authr_right().

Add a right to a GAA_AUTHORIZED credential

Parameters:

cred input/output condition to add right to

right input right

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG cred or right is null, or cred is not a GAA_AUTHORIZED credential.

Note:

If cred is freed with gaa_free_cred, the right will be be freed at the same time.

4.1.1.4 gaa_status gaa_add_cond_eval_callback (gaa_ptr *gaa*, gaa_cond_eval_callback_ptr *cb*, gaa_string_data *type*, gaa_string_data *authority*, int *is_idcred*)

gaa_add_cond_eval_callback().

Add a condition evaluation callback, associated with the specified type and authority.

Parameters:

gaa input/output gaa pointer

cb input condition evaluation callback (should be a callback created with gaa_new_cond_eval_callback()).

type input condition type to associate this callback with

authority input condition authority to associate this callback with

is_idcred input flag – if nonzero, then gaa_inquire_policy_info() will interpret conditions with this type and authority to be identity conditions.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa or cb was null

Note:

When gaa_check_authorization() or gaa_inquire_policy_info() searches for a callback routine for a condition, it first looks for a callback that was installed with the same type and authority as the condition. If no match is found, it searches for a callback with the same authority and a null type. If no match is found, it searches for a callback with the same type and a null authority. If no match is found, it searches for a callback with null type and authority.

4.1.1.5 gaa_status gaa_add_condition (gaa_policy_right * *right*, gaa_condition * *condition*)

gaa_add_condition().

Add a condition to a policy right.

Parameters:

right input right to add

condition input/output condition to add right to.

Return values:

GAA_S_SUCCESS success

4.1.1.6 gaa_status gaa_add_cred (gaa_ptr *gaa*, gaaint_sc * *sc*, gaa_cred * *cred*)

gaa_add_cred().

Add a credential to a security context.

Parameters:

gaa input gaa pointer (ignored).

sc input/output security context.

cred input credential to add

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG sc or cred is null

4.1.1.7 gaa_status gaa_add_cred_condition (gaa_cred * *cred*, gaa_condition * *cond*)

gaa_add_cred_condition().

Add a condition to a credential. This utility function will most likely be used by GAA cred_eval callback functions.

Parameters:

cred input/output credential to add condition to

cond input condition to add.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG cred or cond is null, or the credential is not one of the credential types that accepts conditions.

Note:

If the credential is freed with gaa_free_cred(), the condition will be freed at the same time.

4.1.1.8 gaa_status gaa_add_mech_info (gaa_ptr *gaa*, gaa_string_data *mech_type*, gaa_cred_pull_func *cred_pull*, gaa_cred_eval_func *cred_eval*, gaa_cred_verify_func *cred_verify*, gaa_freefunc *cred_free*, void * *param*, gaa_freefunc *freeparams*)

gaa_add_mech_info().

Create and add a mechinfo callback, which consists of routines to pull additional credentials, evaluate raw credentials, verify credentials, and free raw credentials. This callback can either be associated with a specific mechanism type, or can be installed as a default to be used when no other mechinfo callback matches the requested mechanism type.

Parameters:

gaa input/output gaa pointer

mech_type input mechanism type

cred_pull input cred_pull callback. Used by gaa_pull_creds() to pull additional credentials.

cred_eval input cred_eval callback. Used by gaa_new_cred() to evaluate a raw credential (translate it into a gaa identity, group, etc. credential).

cred_verify input cred_verify callback. Used by gaa_verify_cred() to verify the raw credential (check that it's still valid).

cred_free input cred_free callback. Used by gaa_free_cred() to free the raw credential.

param input mechinfo parameter – passed as an argument to cred_pull, cred_eval, and cred_verify whenever they're called.

freeparam input freeparam function – called to free param when the gaa pointer is freed.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa or mech_type is null

4.1.1.9 gaa_status gaa_add_option (gaa_request_right * *right*, gaa_string_data *type*, gaa_string_data *authority*, void * *value*, gaa_freefunc *freeval*)

gaa_add_option().

Add an option to a request right.

Parameters:

right input/output right

type input option type

authority input option authority

value input option value

freeval optional input function to free value when the option is freed (which will happen automatically when right is freed with gaa_free_request_right().

Return values:

GAA_S_SUCCES success

GAA_S_INVALID_ARG right, t ype, or authority is null.

4.1.1.10 gaa_status gaa_add_policy_entry (gaa_policy * *policy*, gaa_policy_right * *right*, int *priority*, int *num*)

gaa_add_policy_entry().

Add a policy entry to a policy.

Parameters:

policy input/output policy

right input right to add

priority input entry priority

num input entry number (for order within priority)

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG policy or right is null

4.1.1.11 gaa_status gaa_add_policy_entry (gaa_policy * *policy*, gaa_policy_right * *right*, int *priority*, int *num*)

gaa_add_policy_entry().

Add a policy entry to a policy.

Parameters:

policy input/output policy

right input right to add

priority input entry priority

num input entry number (for order within priority)

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG policy or right is null

4.1.1.12 gaa_status gaa_add_request_right (gaa_list_ptr *rightlist*, gaa_request_right * *right*)

gaa_add_request_right().

Add a request right to a list created with gaa_new_req_rightlist().

Parameters:

rightlist input/output list to add right to

right input right to add.

4.1.1.13 gaa_status gaa_check_authorization (gaa_ptr *gaa*, gaa_sc_ptr *sc*, gaa_policy_ptr *policy*, gaa_list_ptr *req_rights*, gaa_answer_ptr *answer*)

gaa_check_authorization().

Check whether the requested rights are authorized under the specified policy.

Parameters:

gaa input gaa pointer

sc input security context

policy input policy

req_rights input list of requested rights

answer output detailed answer – lists all matching policy rights and associated conditions, with flags set to indicate whether each condition was evaluated and/or met. If the result is GAA_C_YES, then the answer includes the time period for which the result is valid (if the start or end time is 0, that time is indefinite). Before being passed to this function, the answer structure should be created with gaa_new_answer().

Return values:

GAA_C_YES Access is granted to all requested rights.

GAA_C_NO Access is denied for at least one requested right.

GAA_C_MAYBE Access is not explicitly denied for any requested right, but there is at least one requested right that GAA cannot decide.

GAA_S_INVALID_ARG sc, policy, answer, or gaa is null

GAA_S_NO_MATCHING_ENTRIES The list of requested rights is empty.

This function makes use of several callback routines – the GAA matchrights callback to determine the subset of the policy that applies to the requested rights, and cond_eval callbacks to evaluate specific conditions. The matchrights callback is also likely to use the valmatch function from the appropriate authinfo callback(s) to determine whether a specific request right matches a specific policy right.

4.1.1.14 gaa_status gaa_check_condition (gaa_ptr *gaa*, gaa_sc_ptr *sc*, gaa_condition * *cond*, gaa_time_period * *vtp*, int * *ynm*, gaa_list_ptr *options*)

gaa_check_condition().

Check a single condition. This utility function is meant to be used in cond_eval callbacks, when evaluating conditions recursively.

Parameters:

gaa input gaa pointer

cond input condition to evaluate

vtp output valid time period

ynm output answer – set to GAA_C_YES, GAA_C_NO, or GAA_C_MAYBE

options optional input request options

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa, sc, cond, or vtp is null

4.1.1.15 gaa_status gaa_check_condition (gaa_ptr *gaa*, gaa_sc_ptr *sc*, gaa_condition * *cond*, gaa_time_period * *vtp*, int * *ynm*, gaa_list_ptr *options*)

gaa_check_condition().

Check a single condition. This utility function is meant to be used in cond_eval callbacks, when evaluating conditions recursively.

Parameters:

gaa input gaa pointer

cond input condition to evaluate

vtp output valid time period

ynm output answer – set to GAA_C_YES, GAA_C_NO, or GAA_C_MAYBE

options optional input request options

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa, sc, cond, or vtp is null

4.1.1.16 void gaa_clear_policy (gaa_policy * *policy*)

gaa_clear_policy().

Clear a policy structure (and free all its entries).

Parameters:

policy input/output policy to clear

4.1.1.17 void gaa_free_answer (gaa_answer * *answer*)

gaa_free_answer().

Free an answer structure.

Parameters:
answer input/output structure to free.

4.1.1.18 void gaa_free_attribute_info (gaa_attribute_info * *info*)

gaa_free_attribute_info.

Free an attribute_info structure and its components.

Parameters:
info input/output attribute info to free

Note:
If a GAA_ATTRIBUTE credential is freed with gaa_free_cred(), this function will be called automatically to free the associated attribute info.

4.1.1.19 void gaa_free_authr_info (gaa_authr_info * *info*)

gaa_free_authr_info().

Free a gaa_authr_info structure (and its components).

Parameters:
info input/output structure to free.

4.1.1.20 void gaa_free_cond_eval_callback (gaa_cond_eval_callback_ptr *cb*)

gaa_free_cond_eval_callback().

Free a condition evaluation callback structure.

Parameters:
cb input/output structure to free.

Note:
If a callback is installed in a gaa structure, then gaa_free() will call this function to free the callback.

4.1.1.21 void gaa_free_condition (gaa_condition * *cond*)

gaa_free_condition().

Free a condition (and all its components).

Parameters:
cond input/output condition to free.

4.1.1.22 void gaa_free_cred (gaa_cred * *cred*)

gaa_free_cred().

Free a credential and its components.

Parameters:
cred input/output credential to free

Note:
This function calls the mechanism-specific cred_free callback function to free the raw credential.

This function is automatically called to free any credential that's part of a security context being freed with gaa_free_sc().

4.1.1.23 void gaa_free_gaa (gaa_ptr *gaa*)

gaa_free_gaa().

Free a gaa structure and its components.

Parameters:
gaa input/output gaa structure to free.

4.1.1.24 void gaa_free_identity_info (gaa_identity_info * *info*)

gaa_free_identity_info().

Free a gaa_identity_info structure (and its components).

Parameters:
info input/output structure to free.

4.1.1.25 void gaa_free_policy (gaa_policy * *policy*)

gaa_free_policy().

Free a policy structure and all its entries.

Parameters:
policy input/output policy to free

4.1.1.26 void gaa_free_policy_entry (gaa_policy_entry * *ent*)

gaa_free_policy_entry().

Free a policy entry and its associated right.

Parameters:
ent input/output entry to free

Note:
If a policy was created using gaa_new_policy() or initialized using gaa_init_policy(), then this function will be called by gaa_free_policy() when the policy is freed.

4.1.1.27 void gaa_free_policy_right (gaa_policy_right * *right*)

gaa_free_policy_right().

Free a policy right.

Parameters:
right input/output right to free

Note:
If a policy was created with gaa_new_policy() or initialized with gaa_init_policy() and is freed with gaa_free_policy(), then this function will be called to free all associated policy rights when the policy is freed.

4.1.1.28 void gaa_free_request_right (gaa_request_right * *right*)

gaa_free_request_right().

Free a request right (and all its components).

Parameters:
right input/output request right to free.

4.1.1.29 void gaa_free_sc (gaa_sc_ptr *sc*)

gaa_free_sc().

Free a gaa security context and its components.

Parameters:
sc input/output security context to free.

4.1.1.30 void gaa_free_sec_attrb (gaa_sec_attrb * *a*)

gaa_free_sec_attrb().

Frees a gaa_sec_attrb.

Parameters:

a input/output sec_attrb to free.

4.1.1.31 void gaa_free_valinfo (gaa_valinfo_ptr *valinfo*)

gaa_free_valinfo.

Free a valinfo structure and its components.

Parameters:

valinfo input/output structure to free

Note:

If a valinfo structure is part of an authinfo structure, then this function will be called automatically to free that valinfo structure when the authinfo structure is freed.

4.1.1.32 char * gaa_get_callback_err ()

gaa_get_err().

Get the gaa thread-specific calback error string.

4.1.1.33 char * gaa_get_err ()

gaa_get_err().

Get the gaa thread-specific error string.

4.1.1.34 gaa_status gaa_get_object_policy_info (gaa_string_data *object*, gaa_ptr *gaa*, gaa_policy_ptr * *policy*)

gaa_get_object_policy_info().

Get policy information for an object. This function calls the installed getpolicy callback.

Parameters:

object input object to get policy for

gaa input gaa pointer

policy output policy to create

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa or policy is null

GAA_S_NO_GETPOLICY_CALLBACK no getpolicy callback was installed.

4.1.1.35 gaa_status gaa_getcreds (gaa_ptr *gaa*, gaa_sc_ptr *sc*, gaa_list_ptr * *credlist*, gaa_cred_type *which*)

gaa_getcreds().

Add credentials of the specified type from the security context to the credential list.

Parameters:

gaa input gaa pointer (ignored)

sc input security context

credlist input/output credential list

which input desired credential type

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG sc or credlist is 0

GAA_S_UNKNOWN_CRED_TYPE which is not a recognized credential type.

4.1.1.36 gaa_status gaa_init_policy (gaa_policy * *policy*)

gaa_init_policy().

Initialize a policy structure.

Parameters:

policy input/output policy to initialize

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG policy is null

4.1.1.37 gaa_status gaa_inquire_policy_info (gaa_ptr *gaa*, gaa_sc_ptr *sc*, gaa_policy_ptr *policy*, gaa_list_ptr * *out_rights*)

gaa_inquire_policy_info().

Return the subset of the input policy that applies to the individual identified with the specified security context. This is the union of the set of rights that do not have any identity conditions with the set of rights whose identity conditions all match the individual.

Parameters:

gaa input gaa pointer

sc input security context

policy input policy

out_rights output list of policy rights

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa, sc, out_rights, or policy is null

Note:

The list returned in out_rights should be freed with gaa_list_free().

4.1.1.38 gaa_status gaa_inquire_policy_info (gaa_ptr *gaa*, gaa_sc_ptr *sc*, gaa_policy_ptr *policy*, gaa_list_ptr * *out_rights*)

gaa_inquire_policy_info().

Return the subset of the input policy that applies to the individual identified with the specified security context. This is the union of the set of rights that do not have any identity conditions with the set of rights whose identity conditions all match the individual.

Parameters:

gaa input gaa pointer

sc input security context

policy input policy

out_rights output list of policy rights

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa, sc, out_rights, or policy is null

Note:

The list returned in out_rights should be freed with gaa_list_free().

4.1.1.39 void * gaa_list_entry_value (gaa_list_entry_ptr *entry*)

gaa_list_entry_value().

Find the data in a list entry.

Parameters:

entry input list entry

Return values:

<***data***> data from list entry

0 entry was null

4.1.1.40 gaa_list_entry_ptr gaa_list_first (gaa_list_ptr *list*)

gaa_list_first().

Find the first entry in a list

Parameters:
list input list

Return values:
<***list_entry***> first list entry
0 list was null

4.1.1.41 void gaa_list_free (gaa_list_ptr *list*)

gaa_list_free().

Free a list and all its entries.

Parameters:
list list to free

Note:
If the list's freefunc is nonzero, it will be called to free the data associated with each list entry.

4.1.1.42 gaa_list_entry_ptr gaa_list_next (gaa_list_entry_ptr *entry*)

gaa_list_next().

Find the next entry in a list

Parameters:
entry input list entry

Return values:
<***list_entry***> next list entry
0 entry was null

4.1.1.43 gaa_status gaa_match_rights (gaa_ptr *gaa*, gaa_request_right * *rright*, gaa_policy_right * *pright*, int * *match*)

gaa_match_rights().

Determines whether a request right matches a policy right. If the two rights do not have the same authority, they don't match. If they do, then the valmatch callback appropriate to that authority is called to determine whether they match or not. This utility function is meant to be used in GAA matchrights callback functions.

Parameters:

gaa input gaa pointer

rright input request right

pright input policy right

match output – set to 1 if they match, 0 if they don't

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa, rright, pright, or match is null

GAA_S_NO_AUTHINFO_CALLBACK No authinfo callback was installed for this authority, and there's no default authinfo callback.

4.1.1.44 gaa_status gaa_match_rights (gaa_ptr *gaa*, gaa_request_right * *rright*, gaa_policy_right * *pright*, int * *match*)

gaa_match_rights().

Determines whether a request right matches a policy right. If the two rights do not have the same authority, they don't match. If they do, then the valmatch callback appropriate to that authority is called to determine whether they match or not. This utility function is meant to be used in GAA matchrights callback functions.

Parameters:

gaa input gaa pointer

rright input request right

pright input policy right

match output – set to 1 if they match, 0 if they don't

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa, rright, pright, or match is null

GAA_S_NO_AUTHINFO_CALLBACK No authinfo callback was installed for this authority, and there's no default authinfo callback.

4.1.1.45 gaa_status gaa_new_answer (gaa_answer ** *answer*)

gaa_new_answer().

Create a new answer structure (suitable for use in a call to gaa_check_authorization()).

Parameters:

answer output answer structure to create

Return values:

GAA_S_SUCCES success

GAA_S_INVALID_ARG answer is null

Note:
A structure created with this function should be freed with gaa_free_answer().

4.1.1.46 gaa_status gaa_new_attribute_info (gaa_ptr *gaa*, gaa_attribute_info ** *info*, gaa_string_data *type*, gaa_string_data *authority*, gaa_string_data *value*)

gaa_new_attribute_info().

Create a new attribute_info structure (to be used as part of a GAA_ATTRIBUTES credential). This utility routine is meant to be used by GAA cred_eval callback functions.

Parameters:
gaa input gaa pointer
info output structure to create
type input attribute type
authority input attribute authority
value input attribute value

Return values:
GAA_S_SUCCESS success
GAA_S_INVALID_ARG info, type, authority, or value is null

Note:
A structure created using this routine should be freed with gaa_free_attribute_info(). This will happen automatically if this structure is part of a credential freed with gaa_free_cred().

4.1.1.47 gaa_status gaa_new_authr_info (gaa_ptr *gaa*, gaa_authr_info ** *info*, void * *objects*, gaa_freefunc *free_objects*)

gaa_new_authr_info().

Create a new gaa_authr_info structure and fill it in with appropriate values.

Parameters:
gaa input gaa pointer
info output structure to create
objects input objects to store in info
free_objects input function to be used to free objects when info is freed.

Return values:
GAA_S_SUCCESS success
GAA_S_INVALID_ARG info or objects is null

Note:
A gaa_authr_info created using this function should be freed with gaa_free_authr_info(). This will happen automatically if it's part of a credential freed with gaa_free_cred().

4.1.1.48 gaa_status gaa_new_cond_eval_callback (gaa_cond_eval_callback_ptr * *cb*, gaa_cond_eval_func *func*, void * *params*, gaa_freefunc *freefunc*)

gaa_new_cond_eval_callback().

Create a condition evaluation callback. If the callback is later installed with gaa_add_cond_eval_callback(), then it will be used (when appropriate) by gaa_check_authorization() and gaa_inquire_policy_info() to evaluate conditions.

Parameters:

cb output callback to create.

func input callback function.

params input callback params – will be passed to func whenever it's called.

freefunc input function to free params when cb is freed.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG cb or func is null.

Note:

A callback created with this function should be freed with gaa_free_cond_eval_callback(). If a callback is added to a gaa structure with gaa_add_cond_eval_callback(), it will be freed automatically when the gaa structure is freed with gaa_free_gaa().

4.1.1.49 gaa_status gaa_new_condition (gaa_condition ** *cond*, gaa_string_data *type*, gaa_string_data *authority*, gaa_string_data *value*)

gaa_new_condition().

Allocates a new gaa_condition structure and fills in the specified values.

Parameters:

cond output condition

type input condition type

authority input condition authority

value input condition value

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG cond is null, or authority is null but value is not.

Note:

Conditions allocated with this function should be freed with gaa_free_condition().

4.1.1.50 gaa_status gaa_new_cred (gaa_ptr *gaa*, gaa_sc_ptr *sc*, gaa_cred ** *cred*, gaa_string_data *mech_type*, void * *mech_spec_cred*, gaa_cred_type *cred_type*, int *evaluate*, gaa_status * *estat*)

gaa_new_cred().

Create a new credential and fill it in with appropriate values.

Parameters:

gaa input gaa pointer

sc input security context

mech_type input credential mechanism type

mech_spec_cred input raw credential

cred_type input credential type (identity, group, etc.).

evaluate input flag – if nonzero, the credential is evaluated (i.e. the appropriate cond_eval callback is called)

estat output – if evaluate and estat are both nonzero, then estat is set to the return value of the cond_eval function.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa, cred, or mech_type is null

GAA_S_UNKNOWN_MECHANISM there are no registered mechanism information callbacks for this mech_type

Note:

A credential created using this function should be freed with gaa_free_cred.

4.1.1.51 gaa_status gaa_new_gaa (gaa_ptr * *gaa*)

gaa_new_gaa().

Create a new gaa pointer.

Parameters:

gaa output gaa pointer to create.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa is null.

Note:

A gaa pointer created using this function should be freed with gaa_free_gaa().

4.1.1.52 gaa_status gaa_new_identity_info (gaa_ptr *gaa*, gaa_identity_info ** *info*)

gaa_new_identity_info().

Create a new gaa_identity_info structure. This is a utility function for use by condition evaluation callback functions.

Parameters:

gaa This argument is ignored.

info output identity info to create.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG info is null.

Note:

A gaa_identity_info created using this function should be freed with gaa_free_identity_info(). This will happen automatically if it's part of a credential freed with gaa_free_cred().

4.1.1.53 gaa_status gaa_new_policy (gaa_policy ** *policy*)

gaa_new_policy().

Create and initialize a policy structure. This utility routine is meant to be used by gaa getpolicy callback functions.

Parameters:

policy output policy to create

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG policy is null

Note:

A policy structure allocated by this function should be freed with gaa_free_policy().

4.1.1.54 gaa_status gaa_new_policy_right (gaa_ptr *gaa*, gaa_policy_right ** *right*, gaa_right_type *type*, gaa_string_data *authority*, gaa_string_data *val*)

gaa_new_policy_right().

Create a new policy right. This utility function is meant to be used by gaa getpolicy callback functions. This function uses the authinfo newval callback to translate the string representation of the value into the appropriate internal format.

Parameters:

gaa input gaa pointer

right output policy right to create

type input right type (pos_access_right or neg_access_right)

authority input right authority

val input string representation of right value

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG right or authority is null

GAA_S_NO_AUTHINFO_CALLBACK no authinfo callback was installed for this authority, and there's no default authinfo callback.

GAA_S_NO_NEWVAL_CALLBACK an authinfo callback was found, but it does not include a newval callback.

4.1.1.55 gaa_status gaa_new_policy_right_rawval (gaa_ptr *gaa*, gaa_policy_right ** *right*, gaa_right_type *type*, gaa_string_data *authority*, void * *val*)

gaa_new_policy_right_rawval().

Allocate a new policy right structure and fill it in with the specified values.

Parameters:

gaa input gaa pointer

right output right pointer

type input right type (pos_access_right or neg_access_right)

authority input authority

val input value

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa, right, or authority is null

GAA_S_NO_AUTHINFO_CALLBACK No authinfo callback was installed appropriate for the specified authority

Note:

Policy rights created with this routine should be freed with gaa_free_policy_right().

This function does not do any translation of the policy right value; the value should be in a form that's understood by the matchrights, copyval, and freeval, and valmatch functions in the authinfo callback associated with this authority.

4.1.1.56 gaa_status gaa_new_policy_right_rawval (gaa_ptr *gaa*, gaa_policy_right ** *right*, gaa_right_type *type*, gaa_string_data *authority*, void * *val*)

gaa_new_policy_right_rawval().

Allocate a new policy right structure and fill it in with the specified values.

Parameters:

gaa input gaa pointer

right output right pointer

type input right type (pos_access_right or neg_access_right)

authority input authority

val input value

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa, right, or authority is null

GAA_S_NO_AUTHINFO_CALLBACK No authinfo callback was installed appropriate for the specified authority

Note:

Policy rights created with this routine should be freed with gaa_free_policy_right().

This function does not do any translation of the policy right value; the value should be in a form that's understood by the matchrights, copyval, and freeval, and valmatch functions in the authinfo callback associated with this authority.

4.1.1.57 gaa_list_ptr gaa_new_req_rightlist ()

gaa_new_req_rightlist().

Create a new list of requested rights. Rights can be added to this list with gaa_add_request_right(), and the result can be used as an argument to gaa_check_authorization().

4.1.1.58 gaa_status gaa_new_request_right (gaa_ptr *gaa*, gaa_request_right ** *right*, gaa_string_data *authority*, gaa_string_data *val*)

gaa_new_request_right().

Allocate a new request right structure and fill it in with the specified values.

Parameters:

right output right pointer

authority input authority

val input string representation of value

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa, right, or authority is null

GAA_S_NO_AUTHINFO_CALLBACK No authinfo callback was installed appropriate for the specified authority

GAA_S_NO_NEWVAL_CALLBACK The authinfo callback does not include a newval function for request rights.

Note:

Request rights created with this routine should be freed with gaa_free_request_right().

This function uses the authinfo callback associated with the specified authority (or the default authinfo callback) to translate the value string into the appropriate representation of the value.

4.1.1.59 gaa_status gaa_new_request_right_rawval (gaa_ptr *gaa*, gaa_request_right ** *right*, gaa_string_data *authority*, void * *value*)

gaa_new_request_right_rawval().

Allocate a new request right structure and fill it in with the specified values.

Parameters:

gaa input gaa pointer

right output right pointer

authority input authority

val input value

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa, right, or authority is null

GAA_S_NO_AUTHINFO_CALLBACK No authinfo callback was installed appropriate for the specified authority

Note:

Request rights created with this routine should be freed with gaa_free_request_right().

This function does not do any translation of the request right value; the value should be in a form that's understood by the matchrights, copyval, and freeval, and valmatch functions in the authinfo callback associated with this authority.

4.1.1.60 gaa_status gaa_new_sc (gaa_sc_ptr * *sc*)

gaa_new_sc().

Create a new gaa security context

Parameters:

sc output security context

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG sc is null.

Note:

A security context created using this function should be freed with gaa_free_sc().

4.1.1.61 gaa_status gaa_new_sec_attrb (gaa_sec_attrb ** *a*, gaa_cred_type *type*, gaa_string_data *authority*, gaa_string_data *value*)

gaa_new_sec_attrb().

Create a new gaa_sec_attrb, and fill it in with the specified values. This is a utility function for use by condition evaluation callback functions.

Parameters:

a output gaa_sec_attrb to create

type input credential type

authority input authority

value input value

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG a is null.

Note:

A gaa_sec_attrb created using this function should be freed with gaa_free_sec_attrb(). This will happen automatically if it's part of a credential freed with gaa_free_cred().

4.1.1.62 gaa_status gaa_new_valinfo (gaa_valinfo_ptr * *valinfo*, gaa_copyval_func *copyval*, gaa_string2val_func *newval*, gaa_freefunc *freeval*, gaa_val2string_func *val2str*)

gaa_new_valinfo().

Allocate a new valinfo structure and fill it in with the specified callback functions.

Parameters:

valinfo output valinfo pointer

copyval input copyval callback function. This callback is used by gaa_check_authorization() and gaa_inquire_policy_info() to create new policy entries.

newval optional input newval callback function. This callback is used by gaa_new_policy_right() and gaa_new_request_right() to translate a string value into the appropriate internal representation.

freeval optional input freeval callback function. This callback is used by gaa_free_request_right() and gaa_free_policy_right() to free right values.

val2str optional input val2str callback function. This callback is used by gaa_request_rightval_string() and gaa_policy_rightval_string() to translate a right value into a string.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG valinfo or copyval is null.

4.1.1.63 gaa_status gaa_new_valinfo (gaa_valinfo_ptr * *valinfo*, gaa_copyval_func *copyval*, gaa_string2val_func *newval*, gaa_freefunc *freeval*, gaa_val2string_func *val2str*)

gaa_new_valinfo().

Allocate a new valinfo structure and fill it in with the specified callback functions.

Parameters:

valinfo output valinfo pointer

copyval input copyval callback function. This callback is used by gaa_check_authorization() and gaa_inquire_policy_info() to create new policy entries.

newval optional input newval callback function. This callback is used by gaa_new_policy_right() and gaa_new_request_right() to translate a string value into the appropriate internal representation.

freeval optional input freeval callback function. This callback is used by gaa_free_request_right() and gaa_free_policy_right() to free right values.

val2str optional input val2str callback function. This callback is used by gaa_request_rightval_string() and gaa_policy_rightval_string() to translate a right value into a string.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG valinfo or copyval is null.

4.1.1.64 char * gaa_policy_rightval_string (gaa_ptr *gaa*, char * *authority*, void * *val*, char * *buf*, int *bsize*)

gaa_policy_rightval_string().

Convert a policy right value and authority into a string. This function calls the policy right val2str callback associated with the specified authority (or the default policy right val2str callback).

Parameters:

gaa input gaa pointer

authority input authority

val input value

buf input buffer – should be large enough to hold the resulting string

bsize input size of buf

Return values:

<***string***> character string representation of the value

0 No authinfo callback was found, or no val2str function was supplied by that callback.

Note:

This function may or may not result in the result string being written into buf, depending on the behavior of the callback function.

4.1.1.65 gaa_status gaa_pull_creds (gaa_ptr *gaa*, gaa_sc_ptr *sc*, gaa_cred_type *which*, gaa_string_data *mech_type*)

gaa_pull_creds().

Locate and call the appropriate callback function to pull additional credentials for the specified mechanism type (or if no mechanism type was specified, call the cred_pull callback functions for all mechanism types), and add the new credentials to the security context.

Parameters:

gaa input gaa pointer

sc input/putput security context

which input what type of credential to pull (identity, group, etc.)

mech_type which mechanism type to pull (or all of them, if 0)

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa or sc is null

GAA_S_UNKNOWN_MECHANISM no mechinfo callback was found for the specified mechanism type

GAA_S_UNKNOWN_MECHANISM a mechinfo callback was found for the specified mechanism type, but it does not include a cred_pull function.

4.1.1.66 char * gaa_request_rightval_string (gaa_ptr *gaa*, char * *authority*, void * *val*, char * *buf*, int *bsize*)

gaa_request_rightval_string().

Convert a request right value and authority into a string. This function calls the request right val2str callback associated with the specified authority (or the default request right val2str callback).

Parameters:

gaa input gaa pointer

authority input authority

val input value

buf input buffer – should be large enough to hold the resulting string

bsize input size of buf

Return values:

<***string***> character string representation of the value

0 No authinfo callback was found, or no val2str function was supplied by that callback.

Note:

This function may or may not result in the result string being written into buf, depending on the behavior of the callback function.

4.1.1.67 gaa_status gaa_set_callback_err (char * *s*)

gaa_get_err().

Set the gaa thread-specific callback error string.

Parameters:

s input string to set the callback error to.

4.1.1.68 gaa_status gaa_set_getpolicy_callback (gaa_ptr *gaa*, gaa_getpolicy_func *func*, void * *param*, gaa_freefunc *freefunc*)

gaa_set_getpolicy_callback().

Set the gaa getpolicy callback. The getpolicy callback function is used by gaa_get_object_policy_info() to create a policy structure containing the policy information associated with an object.

Parameters:

gaa input/output gaa pointer

func input getpolicy function

param input getpolicy parameter (to be passed to func whenever it's called).

freefunc input function to be used to free param when the gaa pointer is freed.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa or func is null

4.1.1.69 gaa_status gaa_set_matchrights_callback (gaa_ptr *gaa*, gaa_matchrights_func *func*, void * *param*, gaa_freefunc *freefunc*)

gaa_set_matchrights_callback().

Set the gaa matchrights callback. This callback is used by gaa_check_authorization() to find the subset of a policy that's relevant to a specific request.

Parameters:

gaa input/output gaa pointer

func input matchrights function

param input getpolicy parameter (to be passed to func whenever it's called).

freefunc input function to be used to free param when the gaa pointer is freed.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG gaa or func is null

4.1.1.70 gaa_status gaa_verify_cred (gaa_cred * *cred*)

gaa_verify_cred().

Calls the appropriate mechanism-specific cred_verify function to verify the credential. This utility routine will most often be used in gaa cond_eval callback functions.

Parameters:

cred input credential to verify

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG cred is null, or no mechanism-specific cred_verify callback was found.

4.2 ”internal gaa routines”

Functions

- gaaint_authinfo* gaa_i_find_authinfo (gaa_ptr gaa, gaa_policy_right * right)
  gaa_i_find_authinfo().
- gaaint_authinfo* gaa_i_auth2authinfo (gaa_ptr gaa, char * authority)
  gaa_i_auth2authinfo().
- void gaa_i_free_authinfo (gaaint_authinfo *ai)
  gaa_i_free_authinfo().
- gaaint_list* gaa_i_new_stack (gaa_freefunc freefunc)
  gaa_i_new_stack().
- gaaint_list* gaa_i_new_silo (gaa_freefunc freefunc)
  gaa_i_new_stack().
- gaaint_list* gaa_i_new_sorted_list (gaa_listcompfunc compare, gaa_freefunc freefunc)
  gaa_i_new_sorted_list().
- gaa_status gaa_i_list_add_entry (gaa_list_ptr list, void *data)
  gaa_i_list_add_entry().
- gaa_status gaa_i_list_add_unique_entry (gaa_list_ptr list, void *data, gaa_listcompfunc checkdups)
  gaa_i_list_add_unique_entry().
- void gaa_i_list_clear (gaaint_list *list)
  gaa_i_list_clear().
- gaa_i_policy_order (gaa_policy_entry *e1, gaa_policy_entry *e2)
  gaa_i_policy_order().
- gaa_i_list_empty (gaaint_list *list)
  gaa_i_list_empty().
- gaa_status gaa_i_list_merge (gaaint_list *dest, gaaint_list *src)
  gaa_i_list_merge().
- gaa_status gaa_i_new_string (char **dest, char *src)
  gaa_i_new_string().
- void gaa_i_free_simple (void *val)
  gaa_i_free_simple().

4.2.1 Function Documentation

4.2.1.1 gaaint_authinfo * gaa_i_auth2authinfo (gaa_ptr *gaa*, char * *authority*)

gaa_i_auth2authinfo().

Find the authinfo callback associated with the specified authority.

Parameters:

gaa input gaa pointer

authority input authority

4.2.1.2 gaaint_authinfo * gaa_i_find_authinfo (gaa_ptr *gaa*, gaa_policy_right * *right*)

gaa_i_find_authinfo().

Find authinfo associated with a policy right

Parameters:

gaa input gaa pointer

right input policy right

4.2.1.3 void gaa_i_free_authinfo (gaaint_authinfo * *ai*)

gaa_i_free_authinfo().

Free an authinfo structure.

Parameters:

ai input/output structure to free

Note:

This function is called by gaa_free_gaa() to free all authinfo callbacks associated with a gaa pointer.

4.2.1.4 void gaa_i_free_simple (void * *val*)

gaa_i_free_simple().

Free a pointer, if it’s nonzero.

Parameters:

val value to free

4.2.1.5 gaa_status gaa_i_list_add_entry (gaa_list_ptr *list*, void * *data*)

gaa_i_list_add_entry().

Add an entry to a list. The position of the new entry will be determined by the list's addfunc.

Parameters:

list input/output list

data input data to add

4.2.1.6 gaa_status gaa_i_list_add_unique_entry (gaa_list_ptr *list*, void * *data*, gaa_listcompfunc *checkdups*)

gaa_i_list_add_unique_entry().

Add an entry to a list, unless an equivalent entry already exists.

Parameters:

list input/output list

data input data to add

checkdups input comparison function to determine whether there's an equivalent entry

4.2.1.7 void gaa_i_list_clear (gaaint_list * *list*)

gaa_i_list_clear().

Clear a list, freeing all its entries

Parameters:

list input/output list to clear

4.2.1.8 gaa_i_list_empty (gaaint_list * *list*)

gaa_i_list_empty().

Check to see whether a list is empty.

Parameters:

list input list to check.

Return values:

0 list is not empty

1 list is empty

4.2.1.9 gaa_status gaa_i_list_merge (gaaint_list * *dest*, gaaint_list * *src*)

gaa_i_list_merge().

Merge two lists into one.

Parameters:

dest input/output list (will be merged list on output)

src input list

4.2.1.10 gaaint_list * gaa_i_new_silo (gaa_freefunc *freefunc*)

gaa_i_new_stack().

Create a new silo.

Parameters:

freefunc optional input function to be used to free list entries when the list is freed.

4.2.1.11 gaaint_list * gaa_i_new_sorted_list (gaa_listcompfunc *compare*, gaa_freefunc *freefunc*)

gaa_i_new_sorted_list().

Create a new sorted list

Parameters:

compare input function to compare list entries (to determine the sort order

freefunc optional input function to be used to free list entries when the list is freed.

4.2.1.12 gaaint_list * gaa_i_new_stack (gaa_freefunc *freefunc*)

gaa_i_new_stack().

Create a new stack.

Parameters:

freefunc optional input function to be used to free list entries when the list is freed.

4.2.1.13 gaa_status gaa_i_new_string (char ** *dest*, char * *src*)

gaa_i_new_string().

Create a new string and copy and old string into it.

Parameters:

dest output string to create

src input string to copy

4.2.1.14 gaa_i_policy_order (gaa_policy_entry * *e1*, gaa_policy_entry * *e2*)

gaa_i_policy_order().

Compare two policy entries. Used in the list of policy entries created by gaa_l_init_policy().

Parameters:

e1 input policy entry to compare

e2 input policy entry to compare

Return values:

-1 e1 < e2 (e1's priority is less than e2's, or the priorities are equal and e1's num is less than e2's)

0 e1 == e2 (the priorities and nums are equal)

1 e2 < e1

4.3 ”gaacore routines”

Functions

- gaacore_has_matchrights_callback (gaa_ptr gaa)
  gaacore_has_matchrights_callback().
- gaacore_has_default_authinfo_callback (gaa_ptr gaa)
  gaacore_has_default_authinfo_callback().
- gaa_status gaacore_set_err (char *s)
  gaacore_set_err().
- char* gaacore_condstat2str (int status)
  gaacore_condstat2str().
- char* gaacore_majstat_str (int status)
  gaacore_majstat_str().
- char* gaacore_right_type_to_string (gaa_right_type rtype)
  gaacore_right_type_to_string().
- char* gaacore_cred_type_to_string (gaa_cred_type ctype)
  gaacore_cred_type_to_string().

4.3.1 Function Documentation

4.3.1.1 char * gaacore_condstat2str (int *status*)

gaacore_condstat2str().

Return a string representation of the condition status.

Parameters:

status input status

4.3.1.2 char * gaacore_cred_type_to_string (gaa_cred_type *ctype*)

gaacore_cred_type_to_string().

Return a string representation of the credential type.

Parameters:
type input credential type

4.3.1.3 gaacore_has_default_authinfo_callback (gaa_ptr *gaa*)

gaacore_has_default_authinfo_callback().

Check whether a default authinfo callback has been set for a gaa pointer.

Parameters:
gaa input gaa to check.

Return values:
1 a default authinfo callback has been set.
0 a default authinfo callback has not been set.

4.3.1.4 gaacore_has_matchrights_callback (gaa_ptr *gaa*)

gaacore_has_matchrights_callback().

Check whether a matchrights callback has been set for a gaa pointer.

Parameters:
gaa input gaa to check.

Return values:
1 a matchrights callback has been set.
0 a matchrights callback has not been set.

4.3.1.5 char * gaacore_majstat_str (int *status*)

gaacore_majstat_str().

Return a string representation of the major part of the status.

Parameters:
status input status

4.3.1.6 char * gaacore_right_type_to_string (gaa_right_type *rtype*)

gaacore_right_type_to_string().

Return a string representation of the right type.

Parameters:
type input right type

4.3.1.7 gaa_status gaacore_set_err (char * s)

gaacore_set_err().

Set the gaa thread-specific error string.

Parameters:

s input string to set the error to.

4.4 ”gaa plugin implementation”

Functions

- gaa_status gaa_plugin_default_matchrights (gaa_ptr gaa, gaa_policy * inpolicy, gaa_request_right *right, gaa_policy * outpolicy, void * params)

  gaa_plugin_default_matchrights().

- gaa_status gaa_plugin_default_new_rval (void **val, char *authority, char *valstr, void *params)

  gaa_plugin_default_new_rval().

- gaa_status gaa_plugin_default_new_pval (void ** val, char * authority, char * valstr, void * params)

  gaa_plugin_default_new_pval().

- gaa_status gaa_plugin_default_copy_pval (void ** newval, char * authority, void * oldval, void * params)

  gaa_plugin_default_copy_pval().

- gaa_status gaa_plugin_default_copy_rval (void ** newval, char * authority, void * oldval, void * params)

  gaa_plugin_default_copy_rval().

- gaa_status gaa_plugin_default_valmatch (char * authority, void * rval, void * pval, void * params)

  gaa_plugin_default_valmatch().

- char* gaa_plugin_default_rval2str (char * authority, void * val, char * buf, int bsize, void * params)

  gaa_plugin_default_rval2str().

- char* gaa_plugin_default_pval2str (char * authority, void * val, char * buf, int bsize, void * params)

  gaa_plugin_default_pval2str().

- void gaa_plugin_default_free_pval (void * pval)

  gaa_plugin_default_free_pval().

- gaa_status gaa_plugin_add_libdir (char * libdir)

  gaa_plugin_add_libdir().

- gaa_status gaa_plugin_install_mechinfo (gaa_ptr gaa, gaa_plugin_mechinfo_args *miargs)

  gaa_plugin_install_mechinfo().

- gaa_status gaa_plugin_install_cond_eval (gaa_ptr gaa, gaa_plugin_cond_eval_args *ceargs)

  gaa_plugin_install_cond_eval().

- gaa_status gaa_plugin_init_mechinfo_args (gaa_plugin_mechinfo_args *miargs)

  gaa_plugin_init_mechinfo_args().

- gaa_status gaa_plugin_init_cond_eval_args (gaa_plugin_cond_eval_args *ceargs)

gaa_plugin_init_cond_eval_args().

- gaa_status gaa_plugin_init_param (gaa_plugin_parameter *param)

  gaa_plugin_init_param().

- gaa_status gaa_plugin_install_authinfo (gaa_ptr gaa, gaa_plugin_authinfo_args *aiargs)

  gaa_plugin_install_authinfo().

- gaa_status gaa_plugin_init_authinfo_args (gaa_plugin_authinfo_args *aiargs)

  gaa_plugin_init_authinfo_args().

- gaa_status gaa_plugin_init_matchrights_args (gaa_plugin_matchrights_args *args)

  gaa_plugin_init_matchrights_args().

- gaa_status gaa_plugin_init_getpolicy_args (gaa_plugin_getpolicy_args *args)

  gaa_plugin_init_getpolicy_args().

- gaa_status gaa_plugin_install_matchrights (gaa_ptr gaa, gaa_plugin_matchrights_args *mrargs)

  gaa_plugin_install_matchrights().

- gaa_status gaa_plugin_install_getpolicy (gaa_ptr gaa, gaa_plugin_getpolicy_args *gpargs)

  gaa_plugin_install_getpolicy().

- gaa_status gaa_plugin_install_mutex_callbacks (gaa_plugin_mutex_args *mxargs)

  gaa_plugin_install_mutex_callbacks().

- gaa_status gaa_plugin_init_mutex_args (gaa_plugin_mutex_args *mxargs)

  gaa_plugin_init_mutex_args().

4.4.1 Function Documentation

4.4.1.1 gaa_status gaa_plugin_add_libdir (char * *libdir*)

gaa_plugin_add_libdir().

Add a directory to the library search path used to find GAA plugin functions.

Parameters:

libdir input directory name.

Return values:

GAA_S_SUCCESS success

4.4.1.2 gaa_status gaa_plugin_default_copy_pval (void ** *newval*, char * *authority*, void * *oldval*, void * *params*)

gaa_plugin_default_copy_pval().

Create a copy of a policy right value. This function assumes that the old policy value was created using gaa_plugin_default_new_pval(), and is intended to be used as a copyval callback in GAA.

Note:

This function allocates space for val; that space should eventually be freed using the gaa_plugin_default_free_pval(). If this function is used as a newval callback in GAA, then gaa_plugin_default_free_pval() should be installed as the corresponding freeval callback.

Parameters:

newval output value pointer.

authority This argument is ignored.

oldval input value to copy

params This argument is ignored.

Return values:

GAA_S_SUCCESS Success

GAA_S_INVALID_ARG One of newval, authority, or oldval is null

4.4.1.3 gaa_status gaa_plugin_default_copy_rval (void ** *newval*, char * *authority*, void * *oldval*, void * *params*)

gaa_plugin_default_copy_rval().

Create a copy of a request right value. This function assumes that the old policy value was created using gaa_plugin_default_new_rval(), and is intended to be used as a copyval callback in GAA.

Note:

This function allocates space for newval; that space should eventually be freed using the standard C free() function. If this function is used as a copyval callback in GAA, then free() should be installed as the corresponding freeval callback.

Parameters:

newval output value pointer.

authority This argument is ignored.

oldval input value to copy

params This argument is ignored.

Return values:

GAA_S_SUCCESS Success

GAA_S_INVALID_ARG One of newval, authority, or oldval is null

4.4.1.4 void gaa_plugin_default_free_pval (void * *pval*)

gaa_plugin_default_free_pval().

Frees a policy value created with gaa_plugin_default_new_pval() or gaa_plugin_default_copy_pval(). Suitable for use as a freeval callback in GAA.

Parameters:

pval policy value to free

4.4.1.5 gaa_status gaa_plugin_default_matchrights (gaa_ptr *gaa*, gaa_policy * *inpolicy*, gaa_request_right * *right*, gaa_policy * *outpolicy*, void * *params*)

gaa_plugin_default_matchrights().

Finds the subset of a policy that matches a request right, and adds that subset to an output policy. This function is intended to be used as a GAA matchrights callback function.

Parameters:

gaa input gaa pointer

inpolicy input policy

right input request right

outpolicy output policy.

params This argument is ignored.

Return values:

GAA_S_SUCCESS Success

GAA_S_INVALID_ARGS One of gaa, inpolicy, right, or outpolicy was null.

4.4.1.6 gaa_status gaa_plugin_default_new_pval (void ** *val*, char * *authority*, char * *valstr*, void * *params*)

gaa_plugin_default_new_pval().

Translate a character string into a policy right value that will be understood by gaa_plugin_default_matchrights(). The character string is interpreted as a comma-separated list of rights. This function is meant to be used as a newval callback in GAA.

Note:

This function allocates space for val; that space should eventually be freed using the gaa_plugin_default_free_pval(). If this function is used as a newval callback in GAA, then gaa_plugin_default_free_pval() should be installed as the corresponding freeval callback.

Parameters:

val output value pointer.

authority This argument is ignored.

valstr input value string

params This argument is ignored.

Return values:

GAA_S_SUCCESS Success

GAA_S_INVALID_ARG One of val, authority, or valstr is null

4.4.1.7 gaa_status gaa_plugin_default_new_rval (void ** *val*, char * *authority*, char * *valstr*, void * *params*)

gaa_plugin_default_new_rval().

Translate a character string into a request right value that will be understood by gaa_plugin_default_matchrights(). This function is meant to be used as a newval callback in GAA.

Note:

This function allocates space for val; that space should eventually be freed using the standard C free() function. If this function is used as a newval callback in GAA, then free() should be installed as the corresponding freeval callback.

Parameters:

val output value pointer.

authority This argument is ignored.

valstr input value string

params This argument is ignored.

Return values:

GAA_S_SUCCESS Success

GAA_S_INVALID_ARG One of val, authority, or valstr is null

4.4.1.8 char * gaa_plugin_default_pval2str (char * *authority*, void * *val*, char * *buf*, int *bsize*, void * *params*)

gaa_plugin_default_pval2str().

Converts a policy right value (created with gaa_plugin_default_new_pval() or gaa_plugin_default_copy_pval()) into a string (a comma-separated list of values). This function is meant to be used as a val2str callback in GAA.

Parameters:

authority This argument is ignored.

val input value

buf output character buffer

bsize input size of buf

params This argument is ignored.

Return values:

The value, represented as a string.

4.4.1.9 char * gaa_plugin_default_rval2str (char * *authority*, void * *val*, char * *buf*, int *bsize*, void * *params*)

gaa_plugin_default_rval2str().

Converts a request right value (created with gaa_plugin_default_new_rval() or gaa_plugin_default_copy_rval()) into a string. This function is meant to be used as a val2str callback in GAA.

Parameters:

authority This argument is ignored.

val input value

buf output character buffer

bsize input size of buf

params This argument is ignored.

Return values:

The value, represented as a string.

4.4.1.10 gaa_status gaa_plugin_default_valmatch (char * *authority*, void * *rval*, void * *pval*, void * *params*)

gaa_plugin_default_valmatch().

Determines whether a request right value (created with gaa_plugin_default_new_rval() or gaa_plugin_default_copy_rval()) matches a policy right value (created with gaa_plugin_default_new_pval() or gaa_plugin_default_copy_pval()); in other words, checks to see whether the request right is in the list of rights that comprise the policy right. This function is intended to be used as a valmatch callback in GAA.

Parameters:

authority This argument is ignored.

rval input request right value

pval input policy right value

params This argument is ignored.

Return values:

1 The request right value matches the policy right value

0 The request right value does not match the policy right value

4.4.1.11 gaa_status gaa_plugin_init_authinfo_args (gaa_plugin_authinfo_args * *aiargs*)

gaa_plugin_init_authinfo_args().

Initializes authinfo args to default values.

Parameters:

aiargs input/output authinfo args.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG aiargs is null

4.4.1.12 gaa_status gaa_plugin_init_cond_eval_args (gaa_plugin_cond_eval_args * *ceargs*)

gaa_plugin_init_cond_eval_args().

Initializes cond_eval args to default values.

Parameters:

ceargs input/output cond_eval args.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG ceargs is null

4.4.1.13 gaa_status gaa_plugin_init_getpolicy_args (gaa_plugin_getpolicy_args * *args*)

gaa_plugin_init_getpolicy_args().

Initializes getpolicy args to default values.

Parameters:

args input/output getpolicy args.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG args is null

4.4.1.14 gaa_status gaa_plugin_init_matchrights_args (gaa_plugin_matchrights_args * *args*)

gaa_plugin_init_matchrights_args().

Initializes matchrights args to default values.

Parameters:

args input/output matchrights args.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG args is null

4.4.1.15 gaa_status gaa_plugin_init_mechinfo_args (gaa_plugin_mechinfo_args * *miargs*)

gaa_plugin_init_mechinfo_args().

Initializes mechinfo args to default values.

Parameters:
miargs input/output mechinfo args.

Return values:
GAA_S_SUCCESS success

GAA_S_INVALID_ARG miargs is null

4.4.1.16 gaa_status gaa_plugin_init_mutex_args (gaa_plugin_mutex_args * *mxargs*)

gaa_plugin_init_mutex_args().

Initializes mutex args to default values.

Parameters:
mxargs input/output mutex args.

Return values:
GAA_S_SUCCESS success

GAA_S_INVALID_ARG mxargs is null

4.4.1.17 gaa_status gaa_plugin_init_param (gaa_plugin_parameter * *param*)

gaa_plugin_init_param().

Initializes parameter args to default values.

Parameters:
param input/output parameter.

Return values:
GAA_S_SUCCESS success

GAA_S_INVALID_ARG param is null

4.4.1.18 gaa_status gaa_plugin_install_authinfo (gaa_ptr *gaa*, gaa_plugin_authinfo_args * *aiargs*)

gaa_plugin_install_authinfo().

Find the appropriate dynamically-linked authinfo routines and parameters, and call gaa_add_authinfo() to install the callback.

Parameters:

gaa input/output gaa pointer

aiargs input authinfo args to install.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG One of gaa, or aiargs was null.

4.4.1.19 gaa_status gaa_plugin_install_cond_eval (gaa_ptr *gaa*, gaa_plugin_cond_eval_args * *ceargs*)

gaa_plugin_install_cond_eval().

Find the appropriate dynamically-linked condition evaluation routines and parameters, and call gaa_add_cond_eval_callback() to install the callback.

Parameters:

gaa input/output gaa pointer

ceargs input cond_eval args to install.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG One of gaa, or ceargs is null

4.4.1.20 gaa_status gaa_plugin_install_getpolicy (gaa_ptr *gaa*, gaa_plugin_getpolicy_args * *gpargs*)

gaa_plugin_install_getpolicy().

Find the appropriate dynamically-linked getpolicy routines and parameters, and call gaa_set_getpolicy_callback() to install the callback.

Parameters:

gaa input/output gaa pointer

gpargs input getpolicy args to install.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG One of gaa or gpargs is null

4.4.1.21 gaa_status gaa_plugin_install_matchrights (gaa_ptr *gaa*, gaa_plugin_matchrights_args * *mrargs*)

gaa_plugin_install_matchrights().

Find the appropriate dynamically-linked matchrights routines and parameters, and call gaa_set_matchrights_callback() to install the callback.

Parameters:

gaa input/output gaa pointer

mrargs input matchrights args to install.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG One of gaa or mrargs is null

4.4.1.22 gaa_status gaa_plugin_install_mechinfo (gaa_ptr *gaa*, gaa_plugin_mechinfo_args * *miargs*)

gaa_plugin_install_mechinfo().

Find the appropriate dynamically-linked mechinfo routines and parameters, and call gaa_add_mech_info() to install the callback.

Parameters:

gaa input/output gaa pointer

miargs input mechinfo args to install.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG One of gaa, miargs, or miargs->mech_type was null.

4.4.1.23 gaa_status gaa_plugin_install_mutex_callbacks (gaa_plugin_mutex_args * *mxargs*)

gaa_plugin_install_mutex_callbacks().

Find the appropriate dynamically-linked mechinfo routines and parameters, and call gaa_add_mech_info() to install the callback.

Parameters:

gaa input/output gaa pointer

mxargs input mutex callback args to install.

Return values:

GAA_S_SUCCESS success

GAA_S_INVALID_ARG One of gaa or mxargs was null.

Chapter 5

References

1. Tatyana Ryutov, Clifford Neuman, Laura Pearlman, ”Generic Authorization and Access control Application Program Interface C-bindings”, Internet-Draft, November 22, 2000
2. Tatyana Ryutov, Clifford Neuman, ”Access Control Framework for Distributed Applications”, Internet-Draft, November 22, 2000
3. Tatyana Ryutov, Clifford Neuman, ”Representation and Evaluation of Security policies for Distributed System Services”, In Proceedings of DARPA Information Survivability Conference & Exposition, January 2000, Hilton Head, South Carolina.
4. Laura Pearlman, GAA-API Source Code
5. Tatyana Ryutov, Original version of GAA-API Source Code

Index