INTERNET-DRAFT
CAT Working Group
Expires April 2001
draft-ietf-cat-gaa-cbind-05.txt

Tatyana Ryutov
Clifford Neuman
Laura Pearlman
USC/Information Sciences Institute
November 22, 2000

Generic Authorization and Access control Application Program Interface C-bindings

0. Status Of this Document

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

To view the entire list of current Internet-Drafts, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), ftp.nordu.net (Northern Europe), ftp.nis.garr.it (Southern Europe), munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu (US West Coast).

1. Abstract

The Generic Authorization and Access control Application Programming Interface (GAA-API) provides access control services to calling applications. It facilitates access control decisions for applications and allows applications to discover access control policies associated with a targeted resource. The GAA-API is usable by multiple applications supporting different kinds of protected objects. The GAA-API design supports:

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

This document specifies C language bindings for the GAA-API, which is described at a language-independent conceptual level in draft-ietf-cat-acc-cntrl-frmw-05.txt

2. GAA-API concepts and typical usage

A simple GAA application will do the following:

a) Perform some initialization at the beginning to create a gaa control structure and security context.

   The gaa control structure (type gaa_ptr) includes information about callback routines (to be used to evaluate conditions, find policy information, etc.). Callback routines may be installed in this structure by the GAA-API implementation itself (when the gaa_ptr structure is created) or explicitly by the application (at any time).

   The security context (type gaa_sc_ptr) contains information about the current user's credentials. Credentials may be added to this structure by the GAA-API implementation itself (in the course of evaluating conditions) or explicitly by the application (at any time).

b) Each time the application receives a request, it will determine what rights are necessary to fulfill that request and then call GAA-API routines to create a list of requested rights, find the relevant policy, and determine whether or not the policy grants those rights.

   Each requested right is of type gaa_request_right, which includes a value and defining authority (the authority determines the namespace of the value). The list of requested rights is created with the gaa_new_req_rightlist function, and rights are added to the list with gaa_add_request_right. A request right may also include a list of options (additional information about the request, to be used as hints when evaluating conditions); the gaa_add_option function may be used to add options to a request right.

   A policy is an ordered list of policy rights. A policy right consists of a type (pos_access_right for a right that's explicitly allowed; neg_access_right for a right that's explicitly denied), a value and defining authority, and a list of conditions under which the policy right applies.

   A condition consists of a type (e.g,. user identity, time of day, etc.), a value and a defining authority (which determines the namespace of the value). In a typical request, the policy will be retrieved with gaa_get_object_policy_info.

   The gaa_check_authorization function is used to determine whether the requested rights are granted or denied by the policy. This function evaluates each requested right (finding the relevant policy rights and

calling the appropriate condition-evaluation callback routines to see whether they apply) and then aggregates the results: if all requested rights are granted, gaa_check_authorization returns GAA_C_YES; if any requested right is denied, it returns GAA_C_NO; otherwise, it returns GAA_C_MAYBE. A detailed answer structure is also returned, including the relevant policy information.

c) When the application is finished using GAA-API, it will call cleanup routines to release resources.

3. GAA-API data types

3.1. Character data

Certain data items used by the GAA-API may be regarded as a character strings, e.g., string-encoded tokens for passing object and authorization database identifiers. The data of this kind is passed between the GAA-API and caller using the gaa_string_data data type, which is a pointer to a null-terminated character array:

```
typedef char *gaa_string_data;
```

3.2. GAA-API status codes

Most GAA-API functions return a value of type gaa_status:

```
unsigned long gaa_status;
```

Encapsulated in the returned status code are major and minor status codes. Each of them has a value range equivalent to 16 bit unsigned integer values. The major code is in low 16 bits, the minor code is in high 16 bits. The major codes indicate errors that are independent of the underlying mechanisms. The errors that can be indicated via a GAA-API major status code are generic API routine errors (errors that are defined in this specification).

The minor code is implementation-dependent and is used to indicate specialized errors from the underlying mechanisms or provide additional information about the GAA-API errors.

A list of GAA-API status codes and their values appears in section 5 of this document.

3.3. Application-opaque data types

3.3.1. The gaa control structure

The gaa control structure (which includes information about the behavior of GAA-API) is opaque to the application:

```
typedef struct gaaint_gaa *gaa_ptr;
```

and is maintained using the callback-registering functions described in section 4.3.

3.3.2. The security context

The gaa security context (which includes information about a user's credentials) is opaque to the application:

```
typedef struct gaaint_sc *gaa_sc_ptr;
```

and is maintained using the credential-handling functions described in section 4.1.2.

3.3.3. List data structures

Certain data items used by the GAA-API may be regarded as an ordered list of data items, e.g., a list of request rights. These lists are represented by the opaque gaa_list_ptr data type; individual entries in these lists are represented by the opaque gaa_list_entry_ptr data type:

```
typedef struct gaaint_list       *gaa_list_ptr;
typedef struct gaaint_list_entry *gaa_list_entry_ptr;
```

The functions described in section 4.6 are used to examine and free lists and list entries.

3.4 Policy data structures

3.4.1 gaa_policy

The gaa_policy structure describes a policy:

```
struct gaa_policy_struct {
   void                *raw_policy;  /* raw policy representation */
   gaa_list_ptr         entries;     /* list of gaa_policy_entry_ptr */
   gaa_freefunc         freeraw;     /* function to free raw_policy */
   struct gaaint_policy *intrl;      /* internal data */
};
typedef struct gaa_policy_struct gaa_policy, *gaa_policy_ptr;
```

The raw_policy field is an application-specific representation of the policy. It may be null, or it may contain, for example, a textual representation of the policy or the address of a policy server to query. The entries field is an ordered list of GAA-API policy rights. If freeraw field is non-null, that function is called to free the raw_policy entry when the policy structure is freed. The intrl field is internal to the GAA-API implementation and should be ignored by the application.

3.4.2 gaa_policy_entry

The gaa_policy_entry structure describes a policy entry:

```
   struct gaa_policy_entry_struct {
      int  priority; /* entry priority */
      int  num;      /* entry number (for order within priority) */
      gaa_policy_right_ptr  right;  /* what this entry grants (or denies) */
   };
   typedef struct gaa_policy_entry_struct  gaa_policy_entry,
                                          *gaa_policy_entry_ptr;
```

The priority and num fields are used to order the entries in a policy (lower-numbered priorities come before higher-numbered priorities; within a priority, lower-numbered entries come before higher-numbered ones). The right field indicates the right that is granted or denied by this policy entry.

3.4.3. gaa_policy_right and gaa_right_type

The gaa_right_type enumeration distinguishes positive policy rights (rights which are explicitly granted) from negative policy rights (rights which are explicitly denied):

```
   typedef enum {
       pos_access_right,
       neg_access_right
   } gaa_right_type;
```

The gaa_policy_right structure describes a policy right:

```
   struct gaa_policy_right_struct {
       gaa_right_type  type;       /* positive or negative */
       gaa_string_data authority;  /* defining authority */
       void           *value;      /* within namespace defined by authority */
       gaa_list_ptr    conditions; /* list of gaa_condition_ptr describing
                                      the conditions that must be met */
       struct gaaint_policy_right *intrl; /* internal data */
   };

   typedef struct gaa_policy_right_struct gaa_policy_right,
                                         *gaa_policy_right_ptr;
```

The type field defines the type of the policy (pos_access_right indicates that the policy grants this right; neg_access_right indicates that the policy denies it).

The authority field indicates the authority responsible for defining the value. The value field is a representation of the value (or values) of the right, within the namespace defined by the authority.

The conditions field is a pointer to an ordered list of elements of type gaa_condition_ptr. It contains a list of pointers to conditions associated with the right.

The intrl field is internal to the GAA-API implementation and should be ignored by the application.

3.4.4. gaa_condition

The gaa_condition structure describes a condition:

```
   struct gaa_condition_struct {
       gaa_string_data type;         /* condition type */
       gaa_string_data authority;    /* defining authority */
       gaa_string_data value;        /* within namespace defined by authority
*/
       unsigned long status;         /* GAA_COND_EVALUATED, GAA_COND_MET, etc.
*/
       struct gaaint_cond *intrl;    /* internal data */
   };
   typedef struct gaa_condition_struct gaa_condition, *gaa_condition_ptr;
```

The type field defines the type of condition (e.g. user_id, time, etc.). The authority field indicates the authority responsible for defining the value (e.g. kerberos, x509, etc.). The value field is the condition value (e.g. a username) within the namespace defined by the authority.

The status field contains flags indicating whether the condition has been evaluated and/or met.

The intrl field is internal to the GAA-API implementation and should be ignored by the application.

3.5 Request rights

Request rights are similar to, but different from, policy rights. Request rights do not have a "type" element (as it makes no sense to request a negative right) or conditions. And while the value of a policy right can represent multiple rights, the value of a request right can represent only a single right.

3.5.1. gaa_request_right

The gaa_request_right structure describes a requested right:

```
   struct gaa_request_right_struct {
       gaa_string_data              authority;/* defining authority */
       void                        *value;    /* within namespace defined
                                                 by authority */
       gaa_list_ptr                 options;  /* list of gaa_request_option */
       struct gaaint_request_right *intrl;    /* internal data */
   };

   typedef struct gaa_request_right_struct gaa_request_right,
                                          *gaa_request_right_ptr;
```

The authority field indicates the authority responsible for defining the value. The value field is a representation of the value of the right, within the namespace defined by the authority.

The options field is a list of elements of type gaa_request_option. It contains a list of pointers to parameters associated with the request right.

The intrl field is internal to the GAA-API implementation and should be ignored by the application.

3.5.2 gaa_request_option

The gaa_request_option stucture is used to provide hints for condition evaluation functions. When an authorization request is made to GAA-API, the condition-evaluation functions are passed a pointer to the requested right, which may contain a list of options. The condition-evaluation function can then look through the list of options to find any that are relevant. For example, if a condition requires that "file size must be less than 10k", then a condition-evaluation function for that condition could check to see whether there was a "file size" option in the request (if there was, the function can return yes or no based on the option's value; if there wasn't, that function must return maybe).

```
struct gaa_request_option {
    gaa_string_data               type;      /* option type */
    gaa_string_data               authority; /* defining authority */
    gaa_string_data               value;     /* within namespace defined
                                                by authority */
    struct gaaint_request_option *intrl;     /* internal data */
};
```

The type field defines the type of the option. The authority field indicates the authority responsible for defining the value, and the value indicates the value of the token (within the namespace defined by the authority field).

The intrl field is internal to the GAA-API implementation and should be ignored by the application.

3.6 Credentials

GAA-API recognizes several types of credentials, which it maintains in a common credential structure.

3.6.1. gaa_cred_type

The gaa_cred_type enumeration describes the different types of credentials:

```
typedef enum  {
```

```
      GAA_IDENTITY,          /* user identity */
      GAA_GROUP_MEMB,        /* group membership */
      GAA_GROUP_NON_MEMB,    /* group non-membership */
      GAA_AUTHORIZED,        /* authorized credential (capability) */
      GAA_ATTRIBUTES,        /* attribute credential */
      GAA_UNEVAL,            /* unevaluated (raw) credential */
      GAA_ANY
    } gaa_cred_type;
```

3.6.2. gaa_principal

The gaa_principal structure describes a principal (an authenticated entity):

```
   struct gaa_principal_struct {
       gaa_cred_type   type;           /* credential type */
       gaa_string_data authority;      /* defining meaning of token value */
       gaa_string_data value;          /* value (in namespace defined
                                          by authority) */
   };
  typedef struct gaa_principal_struct  gaa_principal, *gaa_principal_ptr;
```

The type entry indicates the credential type (identity, group, etc.). The authority field indicates the defining authority (e.g. kerberos, x509), and the value field indicates the value within the namespace defined by the authority.

3.6.3 gaa_cred

The gaa_cred structure is the GAA-API credential data type:

```
   struct gaa_cred_struct {
       gaa_cred_type           type;
       gaa_principal_ptr       grantor;
       gaa_principal_ptr       principal;
       void                   *mech_spec_cred; /* raw credential */
       struct gaaint_mechinfo *mechinfo;    /* functions to handle raw creds */
       union {
          gaa_identity_info_ptr  id_info;
          gaa_authr_info_ptr     authr_info;
          gaa_attribute_ptr      attr_info;
       } info;
   };

   typedef struct gaa_cred_struct  gaa_cred, *gaa_cred_ptr;
```

The type field indicates the type of credential. The grantor field should list the entity that granted the credential. The principal field should list the entity that the credential is for. The mech_spec_cred field is the raw, mechanism-specific credential. The mechinfo field is a pointer to an application-opaque list of

callback functions to be used on this credential and should be ignored by the application. The meaning of the info field depends on the credential type; if the type is GAA_IDENTITY, GAA_GROUP_MEMB, or GAA_GROUP_NON_MEMB, then the id_info field should be filled in. If the type is GAA_AUTHORIZED, then the authr_info field should be filled in. If the type is GAA_ATTRIBUTES, the attr_info field should be filled in.

3.6.4. gaa_identity_info

The gaa_identity_info structure is composed of information specific to identity credentials:

```
  struct gaa_identity_info_struct {
      gaa_list_ptr   conditions;  /* list of gaa_condition_ptr describing
                                     validity constraints */
  };

  typedef struct gaa_identity_info_struct  gaa_identity_info,
                                          *gaa_identity_info_ptr;
```

The conditions field is a pointer to an ordered list of elements of the type gaa_condition_ptr, which lists restrictions placed on the identity, e.g., validity time periods.

Note: the gaa_identity_info structure doesn't contain any "identity" information, because that information is kept in the common area of the gaa_cred structure.

3.6.5. gaa_authr_info

The gaa_authr_info structure contains information specific to authorized credentials (capabilities):

```
  struct gaa_authr_info_struct {
     void                    *objects;
     gaa_list_ptr /* gaa_policy_right_ptr */ access_rights;
     gaa_freefunc free_objects;
  };
  typedef struct gaa_authr_info_struct  gaa_authr_info, *gaa_authr_info_ptr;
```

The objects field is a list of object references to the application-level objects accessible by the grantee, e.g. files or hosts. Object references are from the application-specific name space.

The access_rights field is a pointer to a list of elements of the type gaa_right_ptr. Each element indicate granted or denied access rights.

The free_objects field is a pointer to a function to be called to free the objects field when the gaa_authr_info structure is freed. This field may be 0, in which case no function will be called.

3.6.6 gaa_attribute_info

The gaa_attribute_info structure contains information specific to attribute credentials (credentials that certify that the bearer has some specific attribute):

```
   struct gaa_attribute_info_struct {
       gaa_string_data  type;
       gaa_string_data  authority;
       gaa_string_data  value;
       gaa_list_ptr /* gaa_condition_ptr */  conditions;
   };

   typedef struct gaa_attribute_info_struct  gaa_attribute_info,
                                            *gaa_attribute_info_ptr;
```

The type entry indicates the attribute type (e.g. height, birthdate, etc.). The authority field indicates the defining authority, and the value field indicates the value within the namespace defined by the authority. The conditions field is a list of conditions that must be met in order for the credential to be considered valid.

3.7 Authorization answer structure

The gaa_check_authorization function fills in a detailed answer structure.

3.7.1 gaa_time_period

The gaa_time_period structure describes a time period:

```
   struct gaa_time_period_struct {
      time_t    start_time; /* NULL for unconstrained start time */
      time_t    end_time;   /* NULL for unconstrained end time */
   };
   typedef struct gaa_time_period_struct  gaa_time_period,
                                         *gaa_time_period_ptr;
```

3.7.2 gaa_answer

The gaa_answer structure is the detailed answer from gaa_check_authorization:

```
   struct gaa_answer_struct
   {
      gaa_time_period_ptr  valid_time;
      gaa_list_ptr /* gaa_right_ptr */  rights;
   };
   typedef struct gaa_answer_struct gaa_answer, *gaa_answer_ptr;
```

If the answer was GAA_C_YES, valid_time is the time period for which the answer is valid. The rights field is a list of policy rights that were relevant to the request, with the appropriate evaluated/met flags filled in.

3.8 Function and callback types

GAA-API makes heavy use of callback functions. The GAA-API callback registration model includes three things for each type of callback: a function (or set of related functions), an application-controlled parameter (opaque to the GAA implementation) that is passed to the function(s) each time they are called (to be used to provide operating parameters and/or maintain state information), and a function to be used to free that application-controlled parameter when the application determines that the callback will no longer be needed.

3.8.1 gaa_freefunc

The gaa_freefunc type is used for a function to be used to free arbitrary data.

```
typedef void (*gaa_freefunc)(void *data);
```

3.8.2 Condition-evaluation callbacks

3.8.2.1. gaa_cond_eval_func

```
typedef gaa_status (*gaa_cond_eval_func)(gaa_ptr             gaa,
                                         gaa_sc_ptr          sc,
                                         gaa_condition_ptr   condition,
                                         gaa_time_period_ptr valid_time,
                                         gaa_list_ptr        req_options,
                                         gaa_status          *output_flags,
                                         void                *params);
```

A function of this type should accept the gaa, sc, and condition arguments as input (the condition argument being the condition to evaluate, and the gaa and sc arguments used to find other callback functions and credentials). The req_options argument is an input list of gaa_request_option options that the condition may examine if it chooses to, and the params argument is the optional callback parameter (see gaa_new_cond_eval_callback in section 4.3.1.1).

The output_flags and valid_time arguments are output parameters. The function should set output_flags to the appropriate combination of the GAA_COND_FLG_EVALUATED, GAA_COND_FLG_MET, and GAA_COND_FLG_ENFORCE flags.
It should interpret the valid_time pointer as an output parameter (if the condition imposes time restrictions, the callback function should set the beginning and ending times to whatever the condition restricts them to).

Functions of this type are used by gaa_check_authorization (see section 4.2.3.1), gaa_check_condition (see section 4.4.3.1), and gaa_inquire_policy_info (see section 4.2.3.2).

3.8.2.2. gaa_cond_eval_callback_ptr

```
typedef struct gaaint_cond_eval_callback *gaa_cond_eval_callback_ptr;
```

This is an application-opaque structure to represent a condition evaluation callback.

Note: section 4.3.1 describes functions to create and register condition-evaluation callbacks.

3.8.3 Functions to manipulate raw (mechanism-specific) credentials

3.8.3.1. gaa_cred_pull_func type

```
typedef int (*gaa_cred_pull_func)(gaa_ptr        gaa,
                                  gaa_sc_ptr     sc,
                                  gaa_cred_type  which,
                                  void          *params);
```

A function of this type should pull raw credentials of the type specified by "which" (or all types, if "which" is GAA_ANY) and add them to the security context sc. The function may use the gaa to find other callbacks if appropriate. The params argument is the optional callback-specific parameter (see gaa_add_mech_info in section 4.3.2). Functions of this type are called by gaa_pull_creds (see section 4.4.3.2).

3.8.3.2. gaa_cred_eval_func type

```
typedef int (*gaa_cred_eval_func)(gaa_ptr      gaa,
                                  gaa_sc_ptr   sc,
                                  gaa_cred_ptr cred,
                                  void         *raw,
                                  void         *params);
```

A function of this type should take a raw mechanism-specific credential and fill in the appropriate values in the credential cred (in the process, it should create one of the credential entries in the cred->info union). The gaa and sc arguments are input arguments, and "params" is the optional callback-specific parameter (see gaa_add_mech_info in section 4.3.2). Functions of this type are called by gaa_new_cred (see section 4.1.2.2) to evaluate credentials.

3.8.3.3. gaa_cred_verify_func type

```
typedef int (*gaa_cred_verify_func)(gaa_cred_ptr  cred,
                                    void          *params);
```

A function of this type should take an evaluated gaa credential and verify that it is still valid (i.e. that the raw credential is still valid and still corresponds to the values listed for its grantor, principal, etc.). The "params" argument is the optional callback-specific parameter (see gaa_add_mech_info in section 4.3.2). Functions of this type are used by gaa_verify_cred (see section 4.4.3.3).

3.8.4. Callback to get policy information

```
typedef int (*gaa_getpolicy_func)(gaa_ptr         gaa,
                                  gaa_policy_ptr *policy,
                                  gaa_string_data object,
                                  void           *params);
```

A function of this type should take a gaa pointer, an object name, and an optional pointer to application-specific parameters, and create an output policy structure containing all the policy information that relates to that object.

The getpolicy callback is registered with gaa_set_getpolicy_callback (see section 4.3.3) and used by gaa_get_object_policy_info (see section 4.2.2).

3.8.5 Functions to override GAA default behavior

Each GAA-API implementation has default internal representations of policy right values and request right values and a default function to determine which policy rights match a request right. Most applications can simply use these defaults (and ignore everything in this section); however, GAA-API provides callbacks to override them.

3.8.5.1 Callback to find the subset of a policy that applies to a requested right.

```
typedef int (*gaa_matchrights_func)(gaa_ptr               gaa,
                                    gaa_policy_ptr        inpolicy,
                                    gaa_request_right_ptr right,
                                    gaa_policy_ptr        outpolicy,
                                    void                 *params);
```

A function of this type should take a gaa pointer, an input policy (inpolicy), and an input requested right, and fill in an output policy with those entries from the input policy (in the same order) that apply to the requested right. The gaa_set_matchrights_callback function (see section 4.3.4) is used to set this callback.

3.8.5.2. Function types associated with the internal representation of right values ("valinfo" functions). These callbacks are registered using gaa_add_authinfo (see section 4.3.5).

3.8.5.2.1. gaa_copyval_func

```
typedef gaa_status (*gaa_copyval_func)(void           **newval,
                                       gaa_string_data authority,
                                       void           *oldval,
                                       void           *params);
```

Functions of this type should take a defining authority, a right value

("oldval"), and optional callback parameters ("params"), and create a new value ("newval") that's a duplicate of the original value ("oldval").

3.8.5.2.2. gaa_string2val_func

```
typedef gaa_status (*gaa_string2val_func)(void            **val,
                                          gaa_string_data authority,
                                          gaa_string_data valstr,
                                          void            *params);
```

Functions of this type should take a defining authority, an input string ("valstr"), and optional callback parameters ("params"), and create a new value ("val") containing an internal representation of that string.

3.8.5.2.3. gaa_val2string_func

```
typedef char *(*gaa_val2string_func)(gaa_string_data authority,
                                     void            *val,
                                     gaa_string_data buf,
                                     int             bsize,
                                     void            *params);
```

Functions of this type should take a defining authority, a right value ("val"), a buffer ("buf") of size bsize, and optional callback parameters ("params"), and return a character-string representation of that value. These functions are not required to write the character-string representation into the supplied buffer.

3.8.5.2.4. gaa_valmatch_func

```
typedef int (*gaa_valmatch_func)(gaa_string_data authority,
                                 void            *rval,
                                 void            *pval,
                                 void            *params);
```

Functions of this type should take a defining authority, an input request right ("rval"), and input policy right ("pval"), and optional callback parameters ("params"), and return 1 if the request right matches the policy right and 0 otherwise.

3.8.5.2.5. gaa_valinfo_ptr

The gaa_valinfo_ptr is an application-opaque data type used to register functions of the types described in this section as callbacks:

```
typedef struct gaaint_valinfo *gaa_valinfo_ptr;
```

4. GAA-API functions

Unless otherwise noted, all GAA-API routines return GAA_S_SUCCESS on success and one of the error codes defined in section 5 on failure.

4.1 Initialization functions

4.1.1 gaa_initialize

The gaa_initialize function must be called before any other GAA-API function. It initializes the GAA-API structures and sets up the default behavior of GAA-API routines. (The default behaviors can be modified later using the GAA-API callback registration routines in section 4.3).

```
gaa_status
gaa_initialize(gaa_ptr  *gaa,     /* OUT */
               void     *params  /* IN & OUT, OPTIONAL */);

Parameters:
  gaa
     A pointer to the gaa structure that will be allocated an initialized.

  params
     A handle to an implementation-specific structure, containing
     initialization information. Can be used to return
     implementation-specific output information.
```

A gaa structure created with gaa_initialize should later be freed using gaa_cleanup (see section 4.8).

4.1.2 Routines to keep track of credentials

The security context contains information about credentials. An application will typically create a security context with gaa_new_sc, then create credentials with gaa_new_cred and add them to the security context with gaa_add_cred.

4.1.2.1. gaa_new_sc

The gaa_new_sc routine allocates an empty gaa_sc data structure.

```
gaa_status
gaa_new_sc(gaa_sc_ptr *sc /* OUT */);

Parameters:
  sc
     A pointer to the security context to be allocated.
```

A structure created using this function should be freed using gaa_free_sc (see section 4.8).

4.1.2.2. gaa_new_cred

The gaa_new_cred routine creates a new credential and fills it in with

appropriate values.

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

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.

A credential created using this function should be freed with gaa_free_cred (see section 4.8).

4.1.2.3 gaa_add_cred

The gaa_add_cred routine adds a credential to a security context.

```
gaa_status gaa_add_cred (gaa_ptr       gaa,
                         gaaint_sc_ptr sc,
                         gaa_cred_ptr  cred)
```

Add a credential to a security context.

Parameters:

gaa	input gaa pointer
sc	input/output security context.
cred	input credential to add

4.2. Functions to evaluate an authorization request.

There are three steps to checking an authorization request: creating a list of requested rights to represent the request, finding the policy relevant to the request, and calling a routine to check the requested rights against the policy.

4.2.1. Functions to build the list of requested rights

4.2.1.1. gaa_new_req_rightlist

The gaa_new_req_rightlist function creates an empty list of requested rights:

```
   gaa_list_ptr gaa_new_req_rightlist (int freerights)

   Parameters:
      freerights  input flag to control the behavior of gaa_list_free
                  when (and if) it is called to free this list.  If
                  this flag is 0, then gaa_list_free will free only
                  the list itself, not any of the rights in the list.
                  If freerights is nonzero, then both the list and all
                  the rights contained in it will be freed.

   Return values:
      on success, returns an empty list.
      on failure, returns 0.
```

4.2.1.2. gaa_new_request_right

The gaa_new_request_right function allocates a new request right structure and fill it in with the specified values.

```
   gaa_status gaa_new_request_right (gaa_ptr                gaa,
                                     gaa_request_right_ptr *right,
                                     gaa_string_data        authority,
                                     gaa_string_data        val)

   Parameters:
        right      output right pointer
        authority  input authority
        val        input string representation of value
```

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

Note: some applications that use callbacks to override the GAA-API implementation's default internal representation of right values may wish to use gaa_new_request_right_rawval (see section 4.7.7) instead of this function.

4.2.1.3. gaa_add_option

The gaa_add_option function adds an option to a request right. It may be called several times to add more than one option to a right.

```
   gaa_status gaa_add_option ( gaa_request_right_ptr right,
                               gaa_string_data       type,
                               gaa_string_data       authority,
                               void                 *value,
                               gaa_freefunc          freeval)
```

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().

4.2.1.4. gaa_add_request_right

The gaa_add_request_right function adds a request right to a list.

```
gaa_status gaa_add_request_right (gaa_list_ptr          rightlist,
                                  gaa_request_right_ptr right)
```

Parameters:

rightlist	input/output list to add right to
right	input right to add.

4.2.2. Function to retrieve policy information

The gaa_get_object_policy_info function retrieves policy information for an object. This function calls the installed getpolicy callback.

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

Parameters:

object	input object to get policy for
gaa	input gaa pointer
policy	output policy to create

4.2.3. Functions to make access control decisions.

4.2.3.1. gaa_check_authorization

The gaa_check_authorization function checks whether the requested rights are authorized under the specified policy.

```
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 )
```

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 (see section 4.7.3).

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-API 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.

4.2.3.2. gaa_inquire_policy_info

The gaa_inquire_policy_info function returns 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.

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

Parameters:

gaa	input gaa pointer
sc	input security context
policy	input policy
out_rights	output list of policy rights

The list returned in out_rights should be freed with gaa_list_free (see section 4.6.4).

4.3. Functions to register callbacks with GAA-API.

4.3.1. Functions to register condition-evaluation callbacks.

A condition-evaluation callback is created using gaa_new_cond_eval_callback and associated with a condition type/authority pair using gaa_add_cond_eval_callback.

See section 3.8.2 for descriptions of the data types used by these functions.

4.3.1.1. gaa_new_cond_eval_callback

The gaa_new_cond_eval_callback function creates 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.

```
   gaa_status gaa_new_cond_eval_callback (gaa_cond_eval_callback_ptr  *cb,
                                          gaa_cond_eval_func          func,
                                          void                        params,
                                          gaa_freefunc
freeparams)
```

```
   Parameters:
        cb         output callback to create.
        func       input callback function.
        params     input callback params -- will be passed to func whenever
                   it's called.
        freeparams input function to free params when cb is freed.
```

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().

4.3.1.2. gaa_add_cond_eval_callback

The gaa_add_cond_eval_callback function adds a condition evaluation callback, associated with the specified type and authority.

```
   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)
```

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 (see section 4.2.3.2) will interpret conditions with this type and authority as identity conditions.

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.

4.3.2. Function to register callbacks to deal with mechanism-specific credentials.

The gaa_add_mech_info function creates and adds 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.

See section 3.8.3 for descriptions of the data types used by this function.

```
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                  *params,
                              gaa_freefunc           freeparams)
```

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.
params	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 params when the gaa pointer is freed.

4.3.3. Function to set the callback routine to get object policy information.

The gaa_set_getpolicy_callback function sets the gaa getpolicy callback, which is used by gaa_get_object_policy_info (see section 4.2.2) to create a policy structure containing the policy information associated with an object.

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

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.

4.3.4. Function to override GAA-API's internal function to determine what subset of a policy is relevant to a request.

Each GAA-API implementation has an internal function to compare a list of requested rights with a policy to determine which policy entries are relevant to the request. The gaa_set_matchrights_callback function is used to replace this internal function with one specified by the application. See section 3.8.5 for a description of the gaa_matchrights_func data type.

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

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.

4.3.5. Functions to override the default internal representation of policy right and request right values.

Each GAA-API implementation has internal functions to translate string representations of policy right and request right values into its own internal representation, to compare policy right and request right values, to copy those values, and to express them as character strings. An application may replace those internal functions by using gaa_new_valinfo (to create callback structures consisting of groups of functions) and gaa_add_authinfo (to associate these callback structures with specific authorities).

See section 3.8.5 for descriptions of the data types used in these functions.

4.3.5.1. gaa_new_valinfo

The gaa_new_valinfo function allocates a new valinfo structure and fill it in with the specified callback functions.

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

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.

4.3.5.2. gaa_add_authinfo

The gaa_add_authinfo function adds an authinfo callback. This callback

will be used to interpret and compare policy right values for rights with the specified defining authority.

```
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().

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.

4.4. Functions used primarily within GAA-API callback routines.

4.4.1. Functions used to build credentials (used primarily within mechanism-specific cred_eval callback routines -- see sections 3.8.3.2 and 4.3.2).

4.4.1.1. gaa_new_principal

The gaa_new_principal creates a new gaa_principal (for use within a gaa_cred structure) and fills it in with the specified values.

```
gaa_status gaa_new_principal (gaa_sec_principal_ptr *princ,
                              gaa_cred_type         type,
                              gaa_string_data       authority,
                              gaa_string_data       value)
```

Parameters:

princ	output gaa_principal to create
type	input credential type

```
        authority  input authority
        value      input value
```

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

4.4.1.2. gaa_new_identity_info

The gaa_new_identity_info function creates an identity_info structure (to be used as part of a GAA_IDENTITY, GAA_GROUP_MEMB, or GAA_GROUP_NON_MEMB credential -- see section 3.6).

```
   gaa_status gaa_new_identity_info (gaa_ptr               gaa,
                                     gaa_identity_info_ptr *info)

   Parameters:
        gaa   input
        info  output identity info to create.
```

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().

4.4.1.3. gaa_new_attribute_info

The gaa_new_attribute_info function creates a new attribute_info structure (to be used as part of a GAA_ATTRIBUTES credential -- see section 3.6).

```
gaa_status gaa_new_attribute_info (gaa_ptr gaa,
                                   gaa_attribute_info_ptr *info,
                                   gaa_string_data         type,
                                   gaa_string_data         authority,
                                   gaa_string_data         value)
   Parameters:
        gaa         input gaa pointer
        info        output structure to create
        type        input attribute type
        authority   input attribute authority
        value       input attribute value
```

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().

4.4.1.4. gaa_new_authr_info

The gaa_new_authr_info function creates a new attribute_info structure (to be used as part of a GAA_AUTHORIZED credential -- see section 3.6).

```
gaa_status gaa_new_authr_info (gaa_ptr            gaa,
                               gaa_authr_info_ptr *info,
                               void               *objects,
                               gaa_freefunc       free_objects)
   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.
```

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().

4.4.1.5. gaa_add_authr_right

The gaa_add_authr_right function adds a right to a GAA_AUTHORIZED credential

```
   gaa_status gaa_add_authr_right  (gaa_cred_ptr         cred,
                                    gaa_policy_right_ptr right)

   Parameters:
        cred   input/output condition to add right to
        right  input right
```

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

4.4.1.6 gaa_add_cred_condition

The gaa_add_cred_condition function adds a condition to a credential. The credential must be one of the credential types that accepts conditions (see section 3.6).

```
gaa_status gaa_add_cred_condition (gaa_cred_ptr      cred,
                                  gaa_condition_ptr cond)
   Parameters:
        cred  input/output credential to add condition to
        cond  input condition to add.
```

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

4.4.2. Functions used to build policies (used primarily within gaa_getpolicy callback functions -- see sections 3.8.4 and 4.2.2).

To build a policy, first create it (with gaa_new_policy), then create policy rights (with gaa_new_policy_right, possibly adding conditions with gaa_add_condition) and add them with gaa_add_policy_entry.

4.4.2.1. gaa_new_policy

The gaa_new_policy function creates a new policy structure.

```
   gaa_status gaa_new_policy (gaa_policy_ptr *policy,
                              void           *raw_policy,
                              gaa_freefunc    freeraw)
```

```
   Parameters:
         policy     output policy to create
         raw_policy optional input raw policy
         freeraw    optional input function to free raw_policy when
                    policy is freed.
```

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

4.4.2.2. gaa_new_policy_right

The gaa_new_policy_right function creates a new policy right.

```
   gaa_status gaa_new_policy_right (gaa_ptr              gaa,
                                    gaa_policy_right_ptr *right,
                                    gaa_right_type       type,
                                    gaa_string_data      authority,
                                    gaa_string_data      val)
```

```
   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
```

Note: some applications that use callbacks to override the GAA-API implementation's default internal representation of right values may wish to use gaa_new_policy_right_rawval (see section 4.7.6) instead of this function.

4.4.2.3. gaa_add_condition

The gaa_add_condition function adds a condition to a policy right.

```
   gaa_status gaa_add_condition (gaa_policy_right_ptr right,
                                 gaa_condition_ptr    condition)
```

```
   Parameters:
         right      input right to add
         condition  input/output condition to add right to.
```

4.4.2.4. gaa_add_policy_entry

The gaa_add_policy_entry function adds a policy entry to a policy.

```
gaa_status gaa_add_policy_entry (gaa_policy_ptr       policy,
                                 gaa_policy_right_ptr right,
                                 int                  priority,
                                 int                  num)
```

Parameters:

policy	input/output policy
right	input right to add
priority	input entry priority
num	input entry number (for order within priority)

4.4.3. Functions used primarily in condition-evaluation callbacks.

4.4.3.1. gaa_check_condition

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

```
gaa_status gaa_check_condition (gaa_ptr             gaa,
                                gaa_sc_ptr          sc,
                                gaa_condition_ptr   cond,
                                gaa_time_period_ptr vtp,
                                int                *ynm,
                                gaa_list_ptr        option)
```

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 list (of type gaa_request_option) of request options

4.4.3.2. gaa_pull_creds

The gaa_pull_creds function locates 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.

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

Parameters:

gaa	input gaa pointer
sc	input/output 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)
```

4.4.3.3. gaa_verify_cred

The gaa_verify_cred function calls the appropriate mechanism-specific cred_verify function to verify the credential.

```
   gaa_status gaa_verify_cred (gaa_cred_ptr cred)

   Parameters:
        cred  input credential to verify
```

4.4.3.4. gaa_getcreds

The gaa_getcreds function finds credentials of the specified type in the security context.

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

   Parameters:
        gaa        input gaa pointer
        sc         input security context
        credlist   input/output credential list
        which      input desired credential type
```

4.4.4. Functions for use in gaa_matchrights callback functions.

4.4.4.1. gaa_match_rights

```
   gaa_status gaa_match_rights (gaa_ptr               gaa,
                                gaa_request_right_ptr rright,
                                gaa_policy_right_ptr  pright,
                                int                   *match)
```

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
```

4.4.5. Function for use by all callback functions

```
gaa_status gaa_set_callback_err (gaa_string_data err)

Set the gaa thread-specific callback error string.

Parameters:
      err  input string to set the callback error to.
```

4.5. String functions

These functions return character string representations of values.

4.5.1 gaa_get_err

The gaa_get_err function returns the gaa thread-specific error string.

```
gaa_string_data gaa_get_err ( )
```

4.5.2. gaa_get_callback_err

The gaa_get_callback_err function returns the gaa thread-specific callback error string.

```
gaa_string_data gaa_get_callback_err ( )
```

4.5.3. gaa_request_rightval_string

The gaa_request_rightval_string function converts the value of a request right into a string.

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


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
```

Note: If a val2str callback function was installed for this authority (see section 4.3.5), then that function is used to do the conversion. Calling gaa_request_rightval_string may or may not result in the result string being written into buf, depending on the behavior of the callback function.

4.5.4 gaa_policy_rightval_string

The gaa_policy_rightval_string function converts the value of a policy right into a string.

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

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

Note: If a val2str callback function was installed for this authority (see section 4.3.5), then that function is used to do the conversion. Calling gaa_request_rightval_string may or may not result in the result string being written into buf, depending on the behavior of the callback function.

4.6. List functions

4.6.1. gaa_list_first

The gaa_list_first function finds the first entry in a list.

```
gaa_list_entry_ptr gaa_list_first (gaa_list_ptr list)
```

Parameters:

list	input list

Return values:

<list_entry>	first list entry
0	list was null

4.6.2. gaa_list_next

The gaa_list_next function finds the next entry in a list.

```
gaa_list_entry_ptr gaa_list_next (gaa_list_entry_ptr entry)
```

Parameters:

entry	input list entry

Return values:

<list_entry>	next list entry

```
      0            entry was null
```

4.6.3. gaa_list_entry_value

The gaa_list_entry_value function finds the data in a list entry.

```
   void * gaa_list_entry_value (gaa_list_entry_ptr entry)

   Parameters:
         entry  input list entry

   Return values:
         <data>  data from list entry
         0       entry was null
```

4.6.4. gaa_list_free

The gaa_list_free function frees a list and all its entries.

```
   void gaa_list_free (gaa_list_ptr list)

   Parameters:
         list  list to free

   Note:
        If, when the list was created, a function was specified to free
        the list's entries, that function will be called to free the
        data associated with each list entry.
```

4.7. Miscellaneous functions

4.7.1. gaa_new_condition

The gaa_new_condition function allocates a new gaa_condition structure and fills in the specified values.

```
   gaa_status gaa_new_condition (gaa_condition_ptr *cond,
                                 gaa_string_data   type,
                                 gaa_string_data   authority,
                                 gaa_string_data   value)

   Parameters:
        cond        output condition
        type        input condition type
        authority   input condition authority
        value       input condition value
```

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

4.7.2. gaa_new_gaa

The gaa_new_gaa function creates a new gaa structure.

```
gaa_status gaa_new_gaa (gaa_ptr * gaa)

Parameters:
      gaa  output gaa pointer to create.
```

A gaa structure created using this function should be freed with gaa_free_gaa.

4.7.3. gaa_new_answer

The gaa_new_answer function creates a new answer structure (suitable for use in a call to gaa_check_authorization()).

```
gaa_status gaa_new_answer (gaa_answer_ptr *answer)

Parameters:
      answer  output answer structure to create
```

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

4.7.4. gaa_clear_policy

The gaa_clear_policy function clears a policy structure (and free all its entries).

```
void gaa_clear_policy (gaa_policy_ptr policy)

Parameters:
      policy  input/output policy to clear
```

4.7.5. gaa_init_policy

The gaa_init_policy function initializes a policy structure.

```
gaa_status gaa_init_policy (gaa_policy_ptr policy)

Parameters:
      policy  input/output policy to initialize
```

4.7.6. gaa_new_policy_right_rawval

The gaa_new_policy_right_rawval function is an alternative form of gaa_new_policy_right. It's intended for use by applications that have overriden the default GAA-API internal representation of right values and that wish to set those values directly rather than translating them from character strings.

```
gaa_status gaa_new_policy_right_rawval (gaa_ptr              gaa,
                                        gaa_policy_right_ptr *right,
```

```
                                        gaa_right_type      type,
                                        gaa_string_data     authority,
                                        void                *val)
```

```
   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
```

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

4.7.7. gaa_new_request_right_rawval

The gaa_new_request_right_rawval function is an alternative form of gaa_new_request_right. It's intended for use by applications that have overriden the default GAA-API internal representation of right values and that wish to set those values directly rather than translating them from character strings.

```
   gaa_status gaa_new_request_right_rawval (gaa_ptr               gaa,
                                            gaa_request_right_ptr *right,
                                            gaa_string_data       authority,
                                            void                  *value)
```

```
   Parameters:
        gaa        input gaa pointer
        right      output right pointer
        authority  input authority
        val        input value
```

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

4.8. Functions to release resources.

The functions in this section free GAA-API data structures.

```
   void gaa_free_answer (gaa_answe_ptrr answer)
     Frees an answer structure and its component policy rights.

   void gaa_free_policy (gaa_policy_ptr  policy)
     Frees a policy structure and all its entries.

   void gaa_free_policy_entry (gaa_policy_entry_ptr  ent)
     Frees a policy entry and its associated right.

     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.

```
void gaa_free_policy_right (gaa_policy_right_ptr right)
```
Free a policy right.

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.

```
void gaa_free_cred (gaa_cred_ptr cred)
```
Free a credential and its components.

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().

```
void gaa_free_principal (gaa_principal_ptr princ)
```
Frees a gaa_principal.

Note: If a gaa_principal structure is the principal or grantor in a gaa_cred structure, then this gaa_free_cred will call this function to free that gaa_pricincipal structure when the credential is freed.

```
void gaa_free_attribute_info (gaa_attribute_info_ptr info)
```
Free an attribute_info structure and its components.

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

```
 void gaa_free_authr_info (gaa_authr_info_ptr info)
```
Free a gaa_authr_info structure (and its components).

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

```
void gaa_free_identity_info (gaa_identity_info_ptr info)
```
Free a gaa_identity_info structure (and its components).

Note: If a GAA_IDENTITY, GAA_GROUP_MEMB, or GAA_GROUP_NON_MEMB credential is freed with gaa_free_cred(), this function will be called automatically to free the associated identity info.

```
void gaa_free_condition (gaa_condition_ptr cond)
```
Free a condition (and all its components).

```
void gaa_free_gaa (gaa_ptr gaa)
```

Free a gaa structure and its components.

```
void gaa_free_request_right (gaa_request_right_ptr right)
```
Free a request right (and all its components).

```
void gaa_free_sc (gaa_sc_ptr sc)
```
Free a gaa security context and its components.

```
void gaa_free_cond_eval_callback (gaa_cond_eval_callback_ptr cb)
```
Free a condition evaluation callback structure.

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

```
void gaa_free_valinfo (gaa_valinfo_ptr valinfo)
```
Free a valinfo structure and its components.

Note:
If a valinfo structure is installed in a gaa structure as a callback, then this function will be called automatically to free that valinfo structure when the gaa structure is freed.

```
void gaa_cleanup (gaa_ptr gaa, void *params)
```
Cleans up internal GAA API structures allocated and initialized using the gaa_initialize function. The gaa and params arguments should be the same as those passed to gaa_initialize.

5. Status codes

The GAA-API routines return a status code of type gaa_status.

Encapsulated in the returned status code are major and minor status codes. Each of them has a value range equivalent to 16 bit unsigned integer values. The major code is in low 16 bits, the minor code is in high 16 bits. The major codes indicate errors that are independent of the underlying mechanisms. The errors that can be indicated via a GAA-API major status code are generic API routine errors (errors that are defined in this specification).

The minor code is implementation-dependent and is used to indicate specialized errors from the underlying mechanisms or provide additional information about the GAA-API errors.

GAA_S_SUCCESS	0	Successful completion.
GAA_C_YES	0	An authorization request is granted.
GAA_C_NO	1	An authorization request is denied.
GAA_C_MAYBE	2	An authorization request has not

		been evaluated.
GAA_S_FAILURE	3	The underlying mechanism detected an error for which no specific GAA-API status code is defined.
GAA_S_INVALID_STRING_DATA_HNDL	4	The handle supplied does not point to a valid gaa_string_data structure.
GAA_S_INVALID_LIST_HNDL	5	The handle supplied does not point to a valid gaa_list structure.
GAA_S_INVALID_GAA_HNDL	6	The handle supplied does not point to a valid gaa structure.
GAA_S_INVALID_POLICY_ENTRY_HNDL	7	The handle supplied does not point to a valid gaa_policy_entry structure.
GAA_S_INVALID_POLICY_HNDL	8	The handle supplied does not point to a valid gaa_policy structure.
GAA_S_INVALID_SC_HNDL	9	The handle supplied does not point to a valid gaa_sc structure.
GAA_S_INVALID_ANSWER_HNDL	10	The handle supplied does not point to a valid gaa_answer structure.
GAA_S_INVALID_REQUEST_RIGHT_HNDL	11	The handle supplied does not point to a valid gaa_request_right structure.
GAA_S_INVALID_POLICY_RIGHT_HNDL	12	The handle supplied does not point to a valid gaa_policy_right structure.
GAA_S_INVALID_CONDITION_HNDL	13	The handle supplied does not point to a valid gaa_condition structure.
GAA_S_INVALID_OPTIONS_HNDL	14	The handle supplied does not point to a valid gaa_options structure.
GAA_S_INVALID_IDENTITY_INFO_HNDL	15	The handle supplied does not point to a valid gaa_uneval_cred structure.
GAA_S_INVALID_AUTHR_INFO_HNDL	16	The handle supplied does not point to a valid gaa_authr_cred structure.
GAA_S_INVALID_PRINCIPAL_HNDL	17	The handle supplied does not point to a valid gaa_principal structure.
GAA_S_INVALID_ATTRIBUTE_HNDL	18	The handle supplied does not point to a valid gaa_attribute structure.

GAA_S_UNIMPLEMENTED_FUNCTION	19	The function is not supported by the underlying implementation.
GAA_S_NO_MATCHING_ENTRIES	20	No matching policy entries have been found for the requested right.
GAA_S_POLICY_PARSING_FAILURE	21	Indicates an error during policy parsing.
GAA_S_POLICY_RETRIEVING_FAILURE	22	Indicates an error during policy retrieval process.
GAA_S_INVALID_ARG	23	One or more arguments was invalid.
GAA_S_UNKNOWN_CRED_TYPE	24	The cred_type of a credential is invalid
GAA_S_UNKNOWN_MECHANISM	25	No mechanism-specific callback functions were found for this credential mechanism
GAA_S_NO_CRED_PULL_CALLBACK	26	An attempt was made to pull credentials, but no cred_pull callback had been registered for this mechanism.
GAA_S_NO_AUTHINFO_CALLBACK	27	No authinfo callback has been registered for this authority.
GAA_S_NO_NEWVAL_CALLBACK	28	No newval callback has been registered for this authority.
GAA_S_NO_GETPOLICY_CALLBACK	29	No getpolicy callback has been registered.
GAA_S_NO_MATCHRIGHTS_CALLBACK	30	No matchrights callback has been registered.
GAA_S_INVALID_IDENTITY_CRED	31	The credential's cred_type and principal's cred_type do not match.
GAA_S_BAD_CALLBACK_RETURN	32	A callback routine returned an error.
GAA_S_INTERNAL_ERR	33	There was a GAA internal error.
GAA_S_SYSTEM_ERR	34	There was a system error.
GAA_S_CRED_PULL_FAILURE	35	There was a problem pulling credentials.

```
  GAA_S_CRED_EVAL_FAILURE          36  There was a problem evaluating
                                       credentials

  GAA_S_CRED_VERIFY_FAILURE        37  There was a problem verifying
                                       credentials.

  GAA_S_CONFIG_ERR                 38 There was a configuration error.
```

6. The GAA-API flags

Flags are 32 bits.

Condition flags:

```
  COND_FLG_EVALUATED    0x01  condition has been evaluated
  COND_FLG_MET          0x10  condition has been met
  COND_FLG_ENFORCE      0x100 condition has to be enforced
```

7. The GAA-API usage example

This section provides an example of a simple application which
calls the GAA-API routines.

```
 #include "gaa.h"

  struct my_right {
      char *authority;
      char *value;
  };

  struct my_request {
      char *object;
      struct my_right *my_rights;
  };

  main()
  {
      gaa_ptr gaa = 0;
      void *client_raw_creds;
      char *cred_mechanism;

      gaa_init(&gaa, 0);
          ...
      process_session(gaa, client_raw_creds, cred_mechanism);
          ...
      gaa_cleanup(&gaa, 0);
  }

  /*
   * process_session() -- sample function to process several gaa
   * requests under the same credentials.
```

```
   * Arguments:
   *    gaa - input gaa pointer
   *    client_raw_creds - input raw credentials of client.
   *    cred_mechanism - name of mechanism for client credentials
   *                             (e.g. gss-api).
   * Return values:
   *   0 success
   *   -1 failure
   *
   * This function calls two application-specific functions:
   *   get_my_request, to get a request from the client, and
   *   process_request, to do whatever the request is if
   *   authorization has been granted.
   *
   * Note: this example function doesn't clean up after itself on errors.
   */
  process_session(gaa_ptr gaa, void *client_raw_creds, char *cred_mechanism)
  {
      gaa_status            status;
      gaa_sc_ptr            sc = 0;
      gaa_policy_ptr        policy = 0;
      struct my_request     *myreq;
      struct my_right       *myright;
      gaa_list_ptr          list = 0;
      gaa_cred_ptr          cred = 0;
      gaa_answer_ptr        answer = 0;
      gaa_request_right_ptr right = 0;

      /* First initialize the security context */
      if (gaa_new_sc(&sc) != GAA_S_SUCCESS)
          return(-1);
      if (gaa_new_cred(gaa, sc, &cred, cred_mechanism, client_raw_creds,
                       GAA_IDENTITY, 1, 0) != GAA_S_SUCCESS)
        return(-1);
      if (gaa_add_cred(gaa, sc, cred) != GAA_S_SUCCESS)
        return(-1);

      while (myreq = get_my_request()) {
        /* Find the appropriate policy for the object specified in the request
*/
        if ((status = gaa_get_object_policy_info(myreq->object, gaa,
                                                 &policy)) != GAA_S_SUCCESS)
            return(-1);

        /* Next, build the list of requested rights */
        if ((list = gaa_new_req_rightlist()) == 0)
            return(-1);
        for (myright = myreq->my_rights; myright->value; myright++) {
            if ((status = gaa_new_request_right(gaa, &right, myright-
>authority,
                                                myright->value)) !=
```

```
GAA_S_SUCCESS)
                return(-1);
            if ((status = gaa_add_request_right(list, right)) != GAA_S_SUCCESS)
                return(-1);
        }

        /* Now check to see whether the request is authorized */
        if ((status = gaa_new_answer(&answer)) != GAA_S_SUCCESS)
            return(-1);

        switch (gaa_check_authorization(gaa, sc, policy, list, answer))
        {
        case GAA_C_YES:
            printf("request authorized\n");
            process_request(myreq);
            break;
        case GAA_C_NO:
            printf("request denied\n");
            break;
        case GAA_C_MAYBE:
            printf("request undetermined\n");
            break;
        default:
            fprintf(stderr, "error determining request authorizaton: %s\n",
                    gaa_get_err());
            break;
        }

        /* Finally, clean up after this request. */
        gaa_list_free(list);
        gaa_free_answer(answer);
      }
      gaa_free_sc(sc);
      return(0);
  }
```

8. References

[1] Linn, J., "Generic Security Service Application Program Interface", RFC 1508, Geer Zolot Associate, September 1993.

[2] Wray, "Generic Security Service Application Program Interface V2 - C bindings", Internet draft, May 1997.

[3] T J Hudson, E A Young
SSLeay http://www.livjm.ac.uk/tools/ssleay/

[4] DASCOM Authorization API draft 1.0
http://www.dascom.com

9. Acknowledgments

Carl Kesselman and Douglas Engert have contributed to discussion
of the ideas and material in this specification.

10. Authors' Addresses

Tatyana Ryutov
Clifford Neuman
Laura Pearlman
USC/Information Sciences Institute
4676 Admiralty Way Suite 1001
Marina del Rey, CA 90292-6695
Phone: +1 310 822 1511
E-Mail: {tryutov, bcn, laura}@isi.edu