INTERNET-DRAFT
CAT Working Group
Expires January 2001
draft-ietf-cat-gaa-cbind-04.txt

Tatyana Ryutov
Clifford Neuman
USC/Information Sciences Institute
July 11, 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-04.txt

2. Design approach

We propose a pseudo-object-oriented approach inspired by the programming
style used in [3].
This approach provides an organized representation of the GAA-API concepts and
entities. It defines an encapsulating interface for mapping of particular
GAA-API implementation to the standardized GAA-API entities.
Objects are represented as pointers to pseudo-class structures.
Throughout this draft we use term "class" to refer to a pseudo-class, which is
implemented using C structures and imitate object-oriented design.

2.1. The abstract class

The abstract class describes concepts common across class implementations,
including object creation, initialization, deletion, methods and
attributes.
Concrete class instances define data for attributes and code for class's
methods.
We define three abstract classes: gaa, gaa_policy and gaa_sc.
The general structure of each abstract class is depicted in Figure 1.

2.1.1. The abstract class data structure

The abstract class data structure contains the following fields:

abstract_class_method
The abstract_class_method is a pointer to the abstract_class_method data
structure.
This field is present in each GAA-API class.

abstract class attributes
The abstract class attributes represent a set of data variables, which are
elaborated by the specific instances of a GAA-API class.

```
abstract_class_new(abstract_class_method_ptr method,
                   abstract_class_ptr       *class,
                   gaa_handle_ptr            arglist)
```

The abstract_class_new method creates a new class. Implementation-dependent
information for class creation is supplied using arglist parameter. This method
is present in each GAA-API class.

```
abstract_class_set(abstract_class_method_ptr  method,
                   abstract_class_ptr         class,
                   gaa_handle_ptr             arglist)
```

The abstract_class_set initializes appropriate fields of the
abstract_class_method
structure (attributes and methods).
Implementation-dependent information needed to appropriately initialize the class values is supplied using arglist parameter. This method is present in each GAA-API class.

```
abstract_class_free(abstract_class_ptr class,
                    gaa_handle_ptr     arglist)
```

The abstract_class_free frees the class structure. Depending on the configuration,
this will free the underlying data object. This method is present in each GAA-API
class.

abstract class methods
The abstract class methods are handles to the respective C functions, which implement the remaining class methods. Some of the handles can be NULL if not implemented. The abstract class methods differ across GAA-API classes.

2.1.2. The abstract class method data structure

The abstract class method structure contains the following fields:

type
The type is the numeric type of the abstract class method. This field is present in
each GAA-API class.

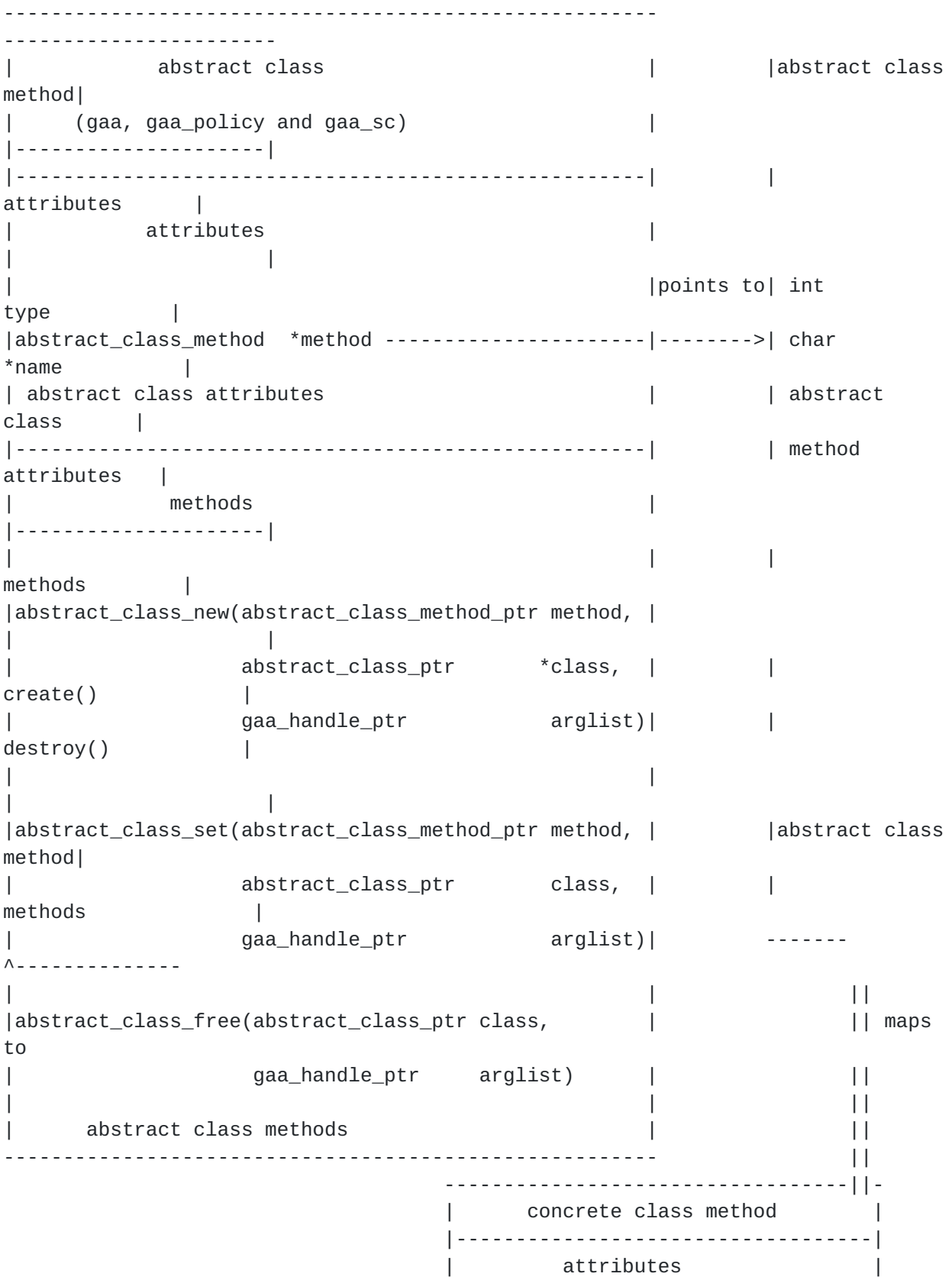
name
The name is a textual representation of the abstract class method of type "type".
This field is present in each GAA-API class.

abstract class method attributes
The abstract class method attributes are handles to the respective C functions, which implement the remaining methods of the abstract class method data structure.
Some of the handles can be NULL if not implemented. The abstract class methods differ across GAA-API class method data structures.

create()
The create method creates a new instance of class method structure of type "type".
This method is present in each GAA-API class method data structure.

destroy()
The destroy frees class method structure of type "type". This method is present in
each GAA-API class method data structure.

The abstract class method methods are handles to the respective C functions, which
implement the methods. Some of them can be NULL if not implemented.



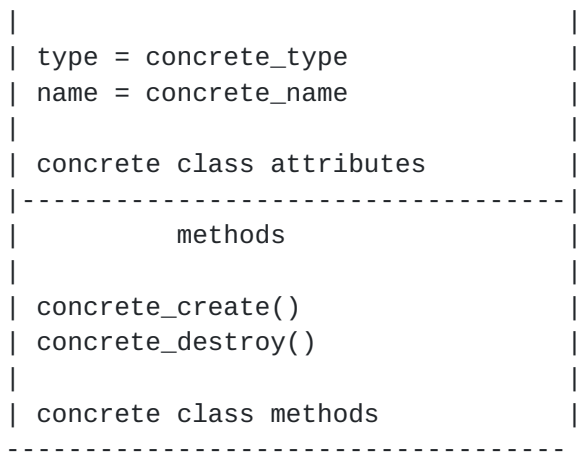


Figure 1.

2.2. Concrete class method

The concrete class method contains concrete values for the abstract class method attributes and methods, which are initialized by the abstract_class_set function.

3. The GAA-API data types and calling conventions

The data types described in this specification refer only to the fields that must be provided by all conforming GAA-API implementations. Individual implementations may provide additional fields for internal use within the GAA-API routines.

3.1. Opaque data

Some data items are considered opaque to the GAA-API, because their internal data structure has no significance to the GAA-API or the calling application, e.g. actual mechanism-specific credentials.

3.1.1. Byte buffer

This type of data is passed between the GAA-API and the caller using a byte buffer referenced by the gaa_buffer_ptr data type, which is a pointer to a gaa_buffer structure.

The gaa_buffer type is a buffer descriptor containing the following fields:

length
The length contains the total number of bytes that the data occupies.

value
The value contains a pointer to the actual data.

```
typedef struct gaa_buffer_struct gaa_buffer,
                                 *gaa_buffer_ptr,
```

```
                                gaa_options,
                               *gaa_options_ptr;

struct gaa_buffer_struct {
size_t     length;
void      *value;
};
```

3.1.2. Implementation-specific data

A number of GAA-API routines need to receive implementation-specific arguments and return implementation-specific values.
The structure of the passed and returned information varies for different implementations, which makes it difficult to standardize the structure.
Data of this type may be regarded as an opaque handle to the implementation-specific data structure and are passed between the GAA-API and caller using the gaa_handle_ptr data structure.

```
unsigned long gaa_handle_ptr
```

Allocation, release and maintenance routines of the underling structure are GAA-API implementation-specific and are not defined in this specification.

3.2. 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
 '\0' terminated C character array:

```
typedef char *gaa_string_data;
```

3.3. Ordered list data

Certain data items used by the GAA-API may be regarded as an ordered list of data items, e.g., a list of identity credentials. Data of this type are passed between the GAA-API and a caller using the gaa_list_ptr data structure.

```
unsigned long gaa_list_ptr
```

List allocation, release and maintenance routines are GAA-API implementation specific and are not defined in this specification.
A possible candidate for implementation of this data type can be STACK structure defined in [3].

3.4. The GAA-API constants

The following constants are used in GAA-API calls and structures:

```
  GAA_C_YES       0 (indicates authorization) is returned if all
```

requested operations are authorized.

GAA_C_NO 1 (indicates denial of authorization) is returned if at least one operation is not authorized.

GAA_C_MAYBE 2 (indicates a need for additional checks) is returned if there are some unevaluated conditions and additional application specific checks are needed, or continuous evaluation is required.

3.5. 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

3.6. Status codes

The GAA-API routines return a status code 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 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_S_FAILURE	3	The underlying mechanism detected an error for which no specific GAA-API status code is defined. The minor code provides details about the error.
GAA_S_INVALID_LIST_HNDL	4	The handle supplied does not point to a valid gaa_list structure.
GAA_S_INVALID_GAA_HNDL	5	The handle supplied does not point to

a
valid gaa structure.

GAA_S_INVALID_GAA_METHOD_HNDL 6 The handle supplied does not point to a
valid gaa_method structure.

GAA_S_INVALID_ANSWER_HNDL 7 The handle supplied does not point to a
valid gaa_answer structure.

GAA_S_INVALID_POLICY_HNDL 10 The handle supplied does not point to a
valid gaa_policy structure.

GAA_S_INVALID_POLICY_METHOD_HNDL 11 The handle supplied does not point to a
valid gaa_policy_method structure.

GAA_S_INVALID_SC_HNDL 12 The handle supplied does not point to a
valid gaa_sc structure.

GAA_S_INVALID_SC_METHOD_HNDL 13 The handle supplied does not point to a
valid gaa_sc_method structure.

GAA_S_INVALID_POLICY_ENTRY_HNDL 15 The handle supplied does not point to a
valid gaa_policy_entry structure

GAA_S_INVALID_CONDITION_HNDL 16 The handle supplied does not point to a
valid gaa_condition structure.

GAA_S_INVALID_RIGHT_HNDL 17 The handle supplied does not point to a
valid gaa_right structure.

GAA_S_INVALID_STRING_DATA_HNDL 18 The handle supplied does not point to a
valid gaa_string_data structure.

GAA_S_INVALID_OPTIONS_HNDL 19 The handle supplied does not point to a
valid gaa_options structure.

GAA_S_INVALID_BUFFER_HNDL 20 The handle supplied does not point to a
valid gaa_buffer structure.

GAA_S_INVALID_ATTRIBUTE_HNDL 21 The handle supplied does not point to

```
a
                                        valid gaa_attribute structure.

  GAA_S_INVALID_AUTHR_CRED_HNDL      22  The handle supplied does not point to
a
                                        valid gaa_authr_cred structure.

  GAA_S_INVALID_UNEVAL_CRED_HNDL     23  The handle supplied does not point to
a
                                        valid gaa_uneval_cred structure.

  GAA_S_UNIMPLEMENTED_FUNCTION       24  The function is not supported by
                                         underlying implementation.

  GAA_S_NO_MATCHING_ENTRIES          25  No matching policy entries have been
                                         found for the requested right.

  GAA_S_POLICY_PARSING_FAILURE       26  Indicates an error during policy
                                         parsing.

  GAA_S_POLICY_RETRIEVING_FAILURE    27  Indicates an error during policy
                                         retrieval process.
```

3.7. GAA-API data structures

3.7.1. gaa_struct data structure

The gaa_struct structure implements the gaa abstract class.
See section 2 for explanation of the meaning of the fields
method, gaa_new, gaa_set and gaa_free.

```
typedef struct gaa_struct  gaa,
                          *gaa_ptr;

struct gaa_struct
{
  /* attributes */

  gaa_method_ptr  method;

  /* methods */

  gaa_status (*gaa_new)(gaa_method_ptr  method,
                        gaa_ptr        *gaa,
                        gaa_handle_ptr  arglist);

  gaa_status (*gaa_set)(gaa_method_ptr  method,
                        gaa_ptr         gaa,
                        gaa_handle_ptr  arglist);

  gaa_status (*gaa_free)(gaa_ptr         gaa,
                         gaa_handle_ptr  arglist);
```

```
};
```

3.7.2. gaa_method_struct data structure

The gaa_method_struct structure implements the gaa abstract class method.
The structure contains information about behavior of the GAA_API evaluation
routines.

The gaa_method_struct structure contains the following fields:

condition_evaluation
The condition_evaluation is a handle to an application-specific condition
evaluation function provided by the calling application. The function is
called by GAA-API if there are application-specific conditions. Generic
(understood by the GAA-API) conditions are evaluated by the GAA-API internal
functions.

calculate_validity_time
The calculate_validity_time is a handle to an implementation-specific function
provided by the calling application. The function is called by the GAA-API to
set the authorization validity time period in the gaa_answer data structure,
see
section 3.8.

See section 2 for explanation of the meaning of the fields type,
name, create and destroy.

```
typedef struct gaa_method_struct  gaa_method,
                                 *gaa_method_ptr;

struct gaa_method_struct
{
 /* attributes */

  int   type;
  char *name;

 /* methods */

  gaa_status (*condition_evaluation)();
  gaa_status (*calculate_validity_time)();
  gaa_status (*create)();
  gaa_status (*destroy)();
};
```

3.7.3. gaa_policy_struct data structure

The gaa_struct structure implements the gaa_policy abstract class.
The gaa_policy_struct structure contains the following fields:

policy
The policy is a pointer to the byte buffer, containing the authorization policy

in application-specific format.

matching_entries
The matching_entries is a pointer to an ordered list of elements of type gaa_policy_entry_ptr returned by the get_matching_entries function, see next section.

See section 2 for explanation of the meaning of the fields method, gaa_policy_new, gaa_policy_set and gaa_policy_free.

```
typedef struct gaa_policy_struct  gaa_policy,
                                 *gaa_policy_ptr;

struct gaa_policy_struct
{
  /* attributes */

  gaa_policy_method_ptr  method;
  gaa_buffer_ptr         policy;
  gaa_list_ptr /* gaa_policy_entry_ptr  */ matching_entries;

  /* methods */

  gaa_status (*gaa_policy_new)(gaa_policy_method_ptr method,
                               gaa_policy_ptr       *policy,
                               gaa_handle_ptr        arglist);

  gaa_status (*gaa_policy_set)(gaa_policy_method_ptr method,
                               gaa_policy_ptr        policy,
                               gaa_handle_ptr        arglist);

  gaa_status (*gaa_policy_free)(gaa_policy_ptr policy,
                                gaa_handle_ptr arglist);

};
```

3.7.4. gaa_policy_method_struct data structure

The gaa_policy_method_struct structure implements the gaa_policy abstract class method. The gaa_policy_method_struct structure contains the following fields:

eval
The eval specifies a policy evaluation approach: based on the order, based on priority or unordered. The default value is the ordered policy evaluation.

get_matching_entries
The get_matching_entries is a handle to an application-specific function for retrieval of the matching entries. The function looks through the policy in application-specific format and finds policies associated with the requested_right.
Then these right-specific policies are translated to the gaa_policy_entry_ptr

and as the result, the function returns an ordered list of elements of type gaa_policy_entry_ptr (see section 3.7.5.), which are then evaluated by the GAA_API
routines.

Return value:

 GAA_S_SUCCESS
 GAA_S_INVALID_POLICY_HNDL
 GAA_S_NO_MATCHING_ENTRIES

retrieve
The retrieve is a handle to an application-specific function for the retrieval of
the object authorization policy. The application maintains authorization information in a form understood by the application. It can be stored in a file,
database, directory service or in some other way. The upcall function provided for
the GAA-API retrieves this information.

Return value:

 GAA_S_SUCCES
 GAA_S_FAILURE
 GAA_S_POLICY_RETRIEVING_FAILURE
 GAA_S_POLICY_PARSING_FAILURE

See section 2 for explanation of the meaning of the fields: type, name, create and destroy.

```
typedef enum  {
 GAA_ORDERED_EVAL  ,
 GAA_PRIORITY_EVAL ,
 GAA_UNORDERED_EVAL
 } gaa_eval_type;


typedef struct gaa_policy_method_struct  gaa_policy_method,
                                        *gaa_policy_method_ptr;

struct gaa_policy_method_struct
{
  /* attributes */

  int            type;
  char          *name;
  gaa_eval_type  eval;

  /* methods */

  gaa_status (*get_matching_entries)(gaa_buffer_ptr policy,           /* IN  */
```

```
                                    gaa_right_ptr  requested_right,  /* IN  */
                                    gaa_list_ptr  *matching_entries  /* OUT
*/);

  gaa_status (*retrieve)(gaa_string_data  object,         /* IN  */
                         gaa_string_data  policy_db,      /* IN  */
                         gaa_buffer_ptr  *buffer, ... );  /* OUT  */

  gaa_status (*create)();
  gaa_status (*destroy)();
};
```

3.7.5. gaa_policy_entry_struct data structure

The gaa_policy_entry_struct structure contains the following fields:

num
The num indicates entry number in the policy. It is used by the GAA_API evaluation routines.

priority
The priority specifies the priority of this entry. It is used by the GAA_API evaluation routines.

rights
The rights is pointer to a linked list of elements of the type gaa_right_ptr. Each element indicates granted or denied access rights.

```
typedef struct gaa_policy_entry_struct  gaa_policy_entry,
                                       *gaa_policy_entry_ptr;

struct gaa_policy_entry_struct {
   int  num;
   int  priority;
   gaa_list_ptr /* gaa_right_ptr */ rights;
};
```

3.7.6. gaa_right_struct data structure

The gaa_right_struct structure contains the following fields:

type
The type defines the type of the token.

authority
The authority indicates the authority responsible for defining the value within the token type.

value
The value indicates the value of the token. The name space for the

value is defined by the authority field.

conditions
The conditions 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.

```
typedef struct gaa_right_struct  gaa_right,
                                *gaa_right_ptr;
struct gaa_right_struct {
   gaa_string_data  type;
   gaa_string_data  authority;
   gaa_string_data  value;
   gaa_list_ptr /* gaa_condition_ptr */ conditions;
};
```

3.7.7. gaa_condition_struct data structure

The gaa_condition_struct structure contains the following fields:

type
The type defines the type of the token.

authority
The authority indicates the authority responsible for defining the
value within the token type.

value
The value indicates the value of the token. The name space for the
value is defined by the authority field.

conditions
The condition 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.

status
The status contains flags, indicating if the condition evaluation status.

```
typedef struct gaa_condition_struct  gaa_condition,
                                    *gaa_condition_ptr;

struct gaa_condition_struct {
   gaa_string_data  type;
   gaa_string_data  authority;
   gaa_string_data  value;
   unsigned long    status;
};
```

3.7.8. gaa_sec_attrb_struct data structure

The gaa_sec_attrb_struct structure contains the following fields:

type
The type defines the type of the token.

authority
The authority indicates the authority responsible for defining the value within the token type.

value
The value indicates the value of the token. The name space for the value is defined by the authority field.

```
struct gaa_sec_attrb_struct {
   gaa_string_data  type;
   gaa_string_data  authority;
   gaa_string_data  value;
};
```

3.7.9. GAA-API Security Context data structures

The gaa_sc_struct structure implements the gaa_sc abstract class, which stores information relevant to access control.

3.7.9.1. gaa_sc_struct data structure

The gaa_sc_struct structure contains the following fields:

sc
The sc is a pointer to a byte buffer, containing the mechanism-specific security context structure.

identity_cred
The identity_cred is a pointer to an ordered list of elements of the type gaa_identity_cred_ptr, containing principal's identity credentials. It is returned by the get_identity_cred function, see next section.

authr_cred
The authr_cred is a pointer to an ordered list of elements of the type gaa_authr_cred_ptr, containing principal's authorization credentials. It is returned by the get_authr_cred function, see next section.

group_membership
The group_membership is a pointer to an ordered list of elements of the type gaa_identity_cred_ptr, which specifies that the grantee is a member of only the listed groups. It is returned by the get_group_membership_cred function,
see next section.

group_non_membership
The group_non_membership is a pointer to an ordered list of elements of

the type gaa_identity_cred_ptr, which specifies that the grantee is NOT a member of the listed groups. It is returned by the get_group_non_membership_cred function, see next section.

attributes
The attributes is a pointer to an ordered list of elements of the type gaa_attribute_ptr, which contains miscellaneous attributes attached to the grantee, e.g., age or security clearance.

uneval_cred
The uneval_cred is a pointer to an ordered list of elements of type gaa_uneval_cred_ptr, containing unevaluated credentials of different types. It is returned by the get_uneval_cred function, see next section.

connection_state
The connection_state is a pointer to a byte buffer, containing a mechanism-specific representation of per-connection context, some of the data stored here include keyblocks and addresses.

See section 2 for explanation of the meaning of the fields method, gaa_sc_new, gaa_sc_set and gaa_sc_free.

```
typedef struct gaa_sc_struct  gaa_sc,
                             *gaa_sc_ptr;
struct gaa_sc_struct
{
 /* attributes */

  gaa_sc_method_ptr  method;
  gaa_buffer_ptr     sc;

  gaa_list_ptr /* gaa_identity_cred_ptr */ identity_cred;
  gaa_list_ptr /* gaa_authr_cred_ptr    */ authr_cred;
  gaa_list_ptr /* gaa_identity_cred_ptr */ group_membership_cred;
  gaa_list_ptr /* gaa_identity_cred_ptr */ group_non_membership_cred;
  gaa_list_ptr /* gaa_attribute_ptr     */ attributes;
  gaa_list_ptr /* gaa_uneval_cred_ptr   */ uneval_cred;

  gaa_buffer_ptr connection_state;

/* methods */

  gaa_status (*gaa_sc_new)(gaa_sc_method_ptr method,
                           gaa_sc_ptr       *sc,
                           gaa_handle_ptr    arglist);

  gaa_status (*gaa_sc_set)(gaa_sc_method_ptr method,
                           gaa_sc_ptr        sc,
                           gaa_handle_ptr    arglist);

  gaa_status (*gaa_sc_free)(gaa_sc_ptr      sc,
```

```
                                gaa_handle_ptr  arglist);
};
```

3.7.9.2. gaa_sc_method_struct data structure

The gaa_sc_method_struct structure implements the gaa_sc abstract class method. The gaa_sc_method_struct structure contains the following fields:

get_identity_cred
The get_identity_cred is a handle to an application-specific function, which translates mechanism-specific credentials to the GAA_API internal structure. It returns an ordered list of elements of type gaa_identity_cred_ptr see section 3.7.9.3, can be NULL if not implemented.

get_authr_cred
The get_authr_cred is a handle to an application-specific function, which translates mechanism-specific credentials to the GAA_API internal structure. It returns an ordered list of elements of type gaa_authr_cred_ptr see section 3.7.9.4, can be NULL if not implemented.

get_group_membership_cred
The get_group_membership_cred is a handle to an application-specific function, which translates mechanism-specific credentials to the GAA-API internal structure.
It returns an ordered list of elements of type gaa_identity_cred_ptr see section 3.7.9.3, can be NULL if not implemented.

get_group_non_membership_cred
The get_group_non_membership_cred is a handle to an application-specific function, which translates mechanism-specific credentials to the GAA API internal structure. It returns an ordered list of elements of type gaa_identity_cred_ptr see section 3.7.9.3, can be NULL if not implemented.

get_attributes
The get_attributes is a handle to an application-specific function, which translates mechanism-specific credentials to the GAA API internal structure. It returns an ordered list of elements of type gaa_attribute_ptr see section 3.7.9.5, can be NULL if not implemented.

get_uneval_cred
The get_uneval_cred is a handle to an application-specific function, which translates mechanism-specific credentials to the GAA-API internal structure. It returns an ordered list of objects of type gaa_uneval_cred_ptr see section 3.7.9.6, can be NULL if not implemented.

pull_cred
The pull_cred is a handle to an application-specific function, which is called when additional credentials are required. It obtains the necessary credentials and
then cred_evaluate function is invoked. This process can be recursive.

cred_evaluate
The cred_evaluate is a handle to an application-specific function, which parses the contents of the acquired credentials into the GAA-API internal form and evaluate them.

See section 2 for explanation of the meaning of the fields type, name, create and destroy.

```
typedef struct gaa_sc_method_struct  gaa_sc_method,
                                    *gaa_sc_method_ptr;

struct gaa_sc_method_struct
{
  /* attributes */

  int   type;
  char *name;

  /* methods */

    gaa_list_ptr /* gaa_identity_cred_ptr */
                (*get_identity_cred)();
    gaa_list_ptr /* gaa_authr_cred_ptr    */
                (*get_authr_cred)();
    gaa_list_ptr /* gaa_identity_cred_ptr */
                (*get_group_membership_cred)();
    gaa_list_ptr /* gaa_identity_cred_ptr */
                (*get_group_non_membership_cred)();
    gaa_list_ptr /* gaa_attribute_ptr     */
                (*get_attributes)();
    gaa_list_ptr /* gaa_uneval_cred_ptr   */
                (*get_uneval_cred)();

   void
   (*condition_evaluation)();

   void
   (*pull_cred)();

   void
   (*cred_evaluate)();

    gaa_status (*create)();
    gaa_status (*destroy)();
};
```

3.7.9.3. gaa_identity_cred_struct data structure

The gaa_identity_cred_struct structure is composed of a set of identity credentials.
Credentials identify the principal on whose behalf the request is performed.

Identity credentials describe a set of mechanism-specific principals, and give their holder the ability to act as any of those principals. Each of the identity credentials contains information needed to authenticate a single principal.

The gaa_identity_cred_struct structure contains the following fields:

principal
The principal identifies an entity on whose behalf the request is performed.

grantor
The grantor identifies an entity who issued the credential.

conditions
The conditions is 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.

mech_spec_cred
The mech_spec_cred is a handle to the actual mechanism-specific identity credential.

```
typedef struct gaa_identity_cred_struct  gaa_identity_cred,
                                        *gaa_identity_cred_ptr;

struct gaa_identity_cred_struct {
   gaa_sec_attrb_ptr   grantor;
   gaa_sec_attrb_ptr   principal;
   gaa_list_ptr /* gaa_condition_ptr */  conditions;
   gaa_buffer_ptr      mech_spec_cred;
};
```

3.7.9.4. gaa_authr_cred_struct data structure

The gaa_authr_cred_struct structure contains the following fields:

grantee
The grantee identifies an entity for whom the credential was issued.

grantor
The grantor identifies an entity who issued the credential.

objects
The object is a pointer to a byte buffer, containing 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.

access_rights
The access_rights is pointer to a linked list of elements of the

type gaa_right_ptr. Each element indicate granted or denied access rights.

conditions
The conditions is a pointer to an ordered list of elements of the type gaa_condition_ptr, which lists restrictions placed on the authorization credential.

mech_spec_cred
The mech_spec_cred is a handle to the actual mechanism-specific authorization credential.

```
typedef struct gaa_authr_cred_struct  gaa_authr_cred,
                                      *gaa_authr_cred_ptr;

struct gaa_authr_cred_struct{
   gaa_sec_attrb_ptr   grantor;
   gaa_sec_attrb_ptr   grantee;
   gaa_buffer          objects;
   gaa_list_ptr /* gaa_right_ptr */ access_rights;
   gaa_buffer_ptr      mech_spec_cred;
};
```

3.7.9.5. gaa_attribute_struct data structure

The gaa_attribute_struct structure contains the following fields:

type
The type defines the type of the token.

authority
The authority indicates the authority responsible for defining the value within the token type.

value
The value indicates the value of the token. The name space for the value is defined by the authority field.

conditions
The conditions is a pointer to an ordered list of elements of the type gaa_condition_ptr, containing pointers to conditions placed on the attribute credential.

mech_spec_cred
Contains a handle to the actual mechanism specific attribute credential

```
typedef struct gaa_attribute_struct  gaa_attribute,
                                     *gaa_attribute_ptr;

struct gaa_attribute_struct {
   gaa_string_data  type;
```

```
   gaa_string_data  authority;
   gaa_string_data  value;
   gaa_list_ptr /* gaa_condition_ptr */  conditions;
   gaa_buffer_ptr   mech_spec_cred;
};
```

3.7.9.6. gaa_uneval_cred_struct data structure

Evaluation of the acquired credentials can be deferred till the credential is actually needed. Unevaluated credentials are stored in the gaa_uneval_cred_struct data structure.

The gaa_uneval_cred_struct structure contains the following fields:

cred_type
Specifies credential type: GAA_IDENTITY, GAA_GROUP_MEMB, GAA_GROUP_NON_MEMB, GAA_AUTHORIZED, and GAA_ATTRIBUTES.

grantee
The grantee identifies an entity for whom the credential was issued.

grantor
The grantor identifies an entity who issued the credential.

mech_type
The mech_type specifies security mechanism used to obtain the credential.

mech_spec_cred
The mech_spec_cred is a handle to the actual mechanism-specific credential.

cred_verification
The cred_verification is a handle to an mechanism-specific credential verification function. It is added to the gaa_uneval_cred structure the by the calling application or transport.

```
typedef enum  {
      GAA_IDENTITY        ,
      GAA_GROUP_MEMB      ,
      GAA_GROUP_NON_MEMB  ,
      GAA_AUTHORIZED      ,
      GAA_ATTRIBUTES
 } gaa_cred_type;

typedef struct gaa_uneval_cred_struct   gaa_uneval_cred,
                                       *gaa_uneval_cred;

struct gaa_uneval_cred_struct {
   gaa_cred_type     cred_type;
   gaa_sec_attrb_ptr grantor;
   gaa_sec_attrb_ptr grantee;
```

```
   gaa_string_data   mech_type;
   gaa_buffer_ptr    mech_spec_cred;
   void (*cred_verification )();
};
```

3.7.10. GAA-API answer data structure

The gaa_check_authorization function returns various information to the application for further evaluation in the gaa_answer data structure.

The gaa_answer_struct structure contains the following fields:

valid_time
The valid_time is a pointer to a structure of type gaa_time_period. It specifies the time period during which the authorization is granted and is returned as a condition to be checked by the application.

rights
The rights is a pointer to an ordered list of structures of the type gaa_right_ptr, which lists granted rights and corresponding conditions, if any.

```
typedef struct gaa_time_period_struct  gaa_time_period,
                                      *gaa_time_period_ptr;
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_answer_struct  gaa_answer,
                                 *gaa_answer_ptr;

struct gaa_answer_struct
{
   gaa_time_period_ptr  valid_time;
   gaa_list_ptr /* gaa_right_ptr */  rights;
};
```

4. GAA-API routine descriptions

This section describes each of the GAA-API routines and discusses their major parameters and how they are to be passed to the routines.

4.1. gaa_initialize routine

Purpose:
The gaa_initialize must be called before any other GAA API function. It initializes the GAA API structures, defines behavior of the gaa evaluation routines.

Parameters:

method
A handle to the implementation-specific gaa method structure, which
implements concrete gaa class, see section 2.

gaa
A handle to the gaa structure.

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

Return value:
         GAA_S_SUCCESS
         GAA_S_FAILURE
         GAA_S_INVALID_GAA_HNDL
         GAA_S_INVALID_GAA_METHOD_HNDL

Synopsis:

```
gaa_status
gaa_initialize(gaa_method_ptr  method,  /* IN  */
               gaa_ptr        *gaa,     /* OUT */
               gaa_handle_ptr  arglist  /* IN & OUT, OPTIONAL */);
```

4.2. gaa_cleanup routine

Purpose:
 The gaa_cleanup cleans up internal GAA API structures allocated and
initialized
 using the gaa_initialize function. The calling application should call
gaa_cleanup
 to free memory and internal implementation state before exiting.

Parameters:

gaa
A handle to a pointer the gaa structure.

arglist
A handle to an implementation-specific structure, containing
clean up information. Can be used to return implementation-specific
output information.

Return value:

       GAA_SUCCESS
       GAA_FAILURE
       GAA_S_INVALID_GAA_HNDL

Synopsis:

```
gaa_status
gaa_cleanup(gaa_ptr        gaa,    /* IN */
            gaa_handle_ptr arglist /* IN & OUT, OPTIONAL */);
```

4.3. gaa_get_object_policy_info routine

Purpose:
The gaa_get_object_policy_info function is called to obtain security policy information associated with the object.

Parameters:

object
Reference to the object to be accessed. The identifier for the object is from an application-specific name space and is opaque to the GAA-API.

policy_db
Reference to an application-specific authorization database, containing access control information for the target object.

policy_handle
A pointer to a handle to gaa_policy structure, containing the security policy associated with the targeted object

Return value:

```
 GAA_S_SUCCESS
 GAA_S_FAILURE
 GAA_S_INVALID_GAA_POLICY_HNDL
 GAA_S_INVALID_GAA_METHOD_HNDL
 GAA_S_UNIMPLEMENTED_FUNCTION
 GAA_S_INVALID_GAA_POLICY_HNDL
 GAA_S_INVALID_GAA_ANSWER_HNDL
```

Synopsis:

```
gaa_status
gaa_get_object_policy_info(gaa_string_data  object,        /* IN  */
                           gaa_string_data  policy_db,     /* IN  */
                           gaa_policy_ptr   policy_handle  /* OUT */)
```

4.4. gaa_check_authorization routine

Purpose:
The gaa_check_authorization function tells the application whether the requested access rights are authorized, or if additional application specific checks are required.

Parameters:

policy_handle
A handle to the gaa_policy structure, returned by the
gaa_get_object_policy_info routine.

gaa
A handle to the gaa structure.

sc
A handle to the principal's security context.

check_access_rights
Ordered list of access rights for authorization.

gaa_options
The optional argument, containing parameters for parameterized operation.

detailed_answer
Contains various information for further evaluation by the application.

Return value:

```
 GAA_YES
 GAA_NO
 GAA_MAYBE
 GAA_S_FAILURE
 GAA_S_INVALID_ACCESS_RIGHTS_HNDL
 GAA_S_INVALID_GAA_POLICY_HNDL
 GAA_S_INVALID_GAA_ANSWER_HNDL
 GAA_S_INVALID_POLICY_METHOD_HNDL
 GAA_S_NO_MATCHING_ENTRIES
 GAA_S_UNIMPLEMENTED_FUNCTION
```

Synopsis:

```
gaa_status
gaa_check_authorization
       (gaa_ptr            gaa,            /* IN&OUT */
        gaa_sc_ptr         sc,             /* IN&OUT */
        gaa_policy_ptr     policy_handle,  /* IN     */
        gaa_options_ptr    gaa_options,    /* IN, OPTIONAL */
        gaa_list_ptr /* gaa_right_ptr */  check_access_rights /* IN  */
        gaa_answer_ptr    *detailed_answer /* OUT    */
       );
```

4.5. gaa_inquire_object_policy_info routine

Purpose:

The gaa_inquire_object_policy_info routine allows calling application
to discover a particular user's rights on an object.

Parameters:

gaa
A handle to the gaa structure.

sc
A handle to the principal's security context.

policy_handle
A handle to the gaa_policy structure, returned by the
gaa_get_object_policy_info routine.

out_rights
A handle to the ordered list of elements of type gaa_right_ptr, which
contains list of rights that the principal is granted or denied.

Return value:

 GAA_SUCCESS
 GAA_FAILURE
 GAA_S_SUCCESS
 GAA_S_FAILURE
 GAA_S_INVALID_ACCESS_RIGHTS_HNDL
 GAA_S_INVALID_GAA_POLICY_HNDL
 GAA_S_INVALID_GAA_ANSWER_HNDL
 GAA_S_INVALID_POLICY_METHOD_HNDL
 GAA_S_UNIMPLEMENTED_FUNCTION
 GAA_S_NO_MATCHING_ENTRIES

Synopsis:

```
gaa_status
gaa_inquire_policy_info
       (gaa_ptr         gaa,           /* IN&OUT */
        gaa_sc_ptr      sc,            /* IN&OUT */
        gaa_policy_ptr  policy_handle, /* IN     */
        gaa_list_ptr   *out_rights     /* OUT    */);
```

5. GAA-API support routines

5.1. Allocation routines

5.1.1. gaa_allocate_buffer routine

Purpose:
Allocate a gaa_buffer data structure and assign default values.

Parameters:

buffer
Pointer to the allocated memory for gaa_buffer structure will be returned.

Return value:

GAA_S_SUCCESS
GAA_S_FAILURE
GAA_S_INVALID_BUFFER_HNDL

Synopsis:

```
gaa_status
gaa_allocate_buffer(gaa_buffer_ptr *buffer /* OUT */);
```

5.1.2. gaa_allocate_answer routine

Purpose:
Allocate a gaa_answer data structure and assign default values.

Parameters:

buffer
A handle to the allocated memory for gaa_answer structure will be returned.

Return value:

GAA_S_SUCCESS
GAA_S_FAILURE
GAA_S_INVALID_ANSWER_HNDL

Synopsis:

```
gaa_status
gaa_allocate_answer(gaa_answer_ptr *buffer /* OUT */);
```

5.1.3. gaa_allocate_condition routine

Purpose:
Allocate a gaa_condition data structure and assign default values.

Parameters:

buffer
A handle to the allocated memory for gaa_condition structure will be returned.

Return value:

GAA_S_SUCCESS
GAA_S_FAILURE
GAA_S_INVALID_CONDITION_HNDL

Synopsis:

```
gaa_status
gaa_allocate_condition(gaa_condition_ptr *buffer /* OUT */);
```

5.1.4. gaa_allocate_right routine

Purpose:
Allocate a gaa_right data structure and assign default values.

Parameters:

buffer
A handle to the allocated memory for gaa_right structure will be returned.

Return value:

GAA_S_SUCCESS
GAA_S_FAILURE
GAA_S_INVALID_RIGHT_HNDL

Synopsis:

```
gaa_status
gaa_allocate_right (gaa_right_ptr *buffer /* OUT */);
```

5.1.5. gaa_allocate_sec_attrb routine

Purpose:
Allocate a gaa_sec_attrb data structure and assign default values.

Parameters:

buffer
A handle to the allocated memory for gaa_sec_attrb structure will be returned.

Return value:

GAA_S_SUCCESS
GAA_S_FAILURE
GAA_S_INVALID_SEC_ATTRB_HNDL

Synopsis:

```
gaa_status
gaa_allocate_sec_attrb (gaa_sec_attrb_ptr *buffer /* IN  */);
```

5.1.6. gaa_allocate_identity_cred routine

Purpose:
Allocate a gaa_identity_cred data structure and assign default values.

Parameters:

buffer
A handle to the allocated memory for gaa_identity_cred structure will be returned.

Return value:

GAA_S_SUCCESS
GAA_S_FAILURE
GAA_S_INVALID_IDENTITY_CRED_HNDL

Synopsis:

```
gaa_status
gaa_allocate_identity_cred(gaa_identity_cred_ptr *buffer /* OUT */);
```

5.1.7. gaa_allocate_authr_cred routine

Purpose:
Allocate a gaa_authr_cred data structure and assign default values.

Parameters:

buffer
A handle to the allocated memory for gaa_authr_cred structure will be returned.

Return value:

GAA_S_SUCCESS
GAA_S_FAILURE
GAA_S_INVALID_AUTHR_CRED_HNDL

Synopsis:

```
gaa_status
gaa_allocate_authr_cred(gaa_authr_cred_ptr *buffer /* OUT */);
```

5.1.8. gaa_allocate_uneval_cred routine

Purpose:
Allocate a gaa_uneval_cred data structure and assign default values.

Parameters:

buffer
A handle to the allocated memory for gaa_uneval_cred structure will be returned.

Return value:

GAA_S_SUCCESS
GAA_S_FAILURE
GAA_S_INVALID_UNEVAL_CRED_HNDL

Synopsis:

```
gaa_status
gaa_allocate_uneval_cred(gaa_uneval_cred_ptr *buffer /* OUT */);
```

5.1.9. gaa_allocate_attribute routine

Purpose:
Allocate a gaa_attribute data structure and assign default values.

Parameters:

buffer
A handle to the allocated memory for gaa_attribute structure will be returned.

Return value:

GAA_S_SUCCESS
GAA_S_FAILURE
GAA_S_INVALID_ATTRIBUTE_HNDL

Synopsis:

```
gaa_status
gaa_allocate_attribute_cred(gaa_attribute_ptr *buffer /* OUT */);
```

5.1.10. gaa_allocate_policy_entry routine

Purpose:
Allocate a gaa_policy_entry data structure and assign default values.

Parameters:

buffer
A handle to the allocated memory for gaa_policy_entry structure will be returned.

Return value:

GAA_S_SUCCESS
GAA_S_FAILURE
GAA_S_INVALID_POLICY_ENTRY_HNDL

Synopsis:

```
gaa_status
gaa_allocate_policy_entry(gaa_policy_entry_ptr *buffer /* OUT */);
```

5.2. Release routines

5.2.1. gaa_free_buffer routine

Purpose:
Free storage associated with a buffer.

Parameters:

buffer
The storage associated with the buffer will be freed.

Return value:
none

Synopsis:

```
void
gaa_free_buffer(gaa_answer_ptr buffer /* IN */);
```

5.2.2. gaa_free_answer routine

Purpose:
Free storage associated with a buffer.

Parameters:

buffer
The storage associated with the buffer will be freed.

Return value:
none

Synopsis:

```
void
gaa_free_answer(gaa_answer_ptr buffer/* IN */);
```

5.2.3. gaa_free_policy_entry routine

Purpose:
Free storage associated with a buffer.

Parameters:

buffer
The storage associated with the buffer will be freed.

Return value:
none

Synopsis:

```
void
gaa_free_policy_entry (gaa_policy_entry_ptr buffer/* IN */);
```

5.2.4. gaa_free_identity_cred routine

Purpose:
Free storage associated with a buffer.

Parameters:

buffer
The storage associated with the buffer will be freed.

Return value:
none

Synopsis:

```
void
gaa_free_identity_cred(gaa_identity_cred_ptr buffer /* IN */);
```

5.2.5. gaa_free_right routine

Purpose:
Free storage associated with a buffer.

Parameters:

buffer
The storage associated with the buffer will be freed.

Return value:
none

Synopsis:

```
void
gaa_free_right (gaa_right_ptr buffer /* IN */);
```

5.2.6. gaa_free_condition routine

Purpose:
Free storage associated with a buffer.

Parameters:

buffer
The storage associated with the buffer will be freed.

Return value:

none

Synopsis:

```
void
gaa_free_condition(gaa_condition_ptr condition /* IN */);
```

5.2.7. gaa_free_sec_attrb routine

Purpose:
Free storage associated with a buffer.

Parameters:

buffer
The storage associated with the buffer will be freed.

Return value:
none

Synopsis:

```
void
gaa_free_sec_attrb (gaa_sec_attrb_ptr buffer /* IN */);
```

5.2.8. gaa_free_authr_cred routine

Purpose:
Free storage associated with a buffer.

Parameters:

buffer
The storage associated with the buffer will be freed.

Return value:
none

Synopsis:

```
void
gaa_free_authr_cred(gaa_authr_cred_ptr buffer /* IN */);
```

5.2.9. gaa_free_uneval_cred routine

Purpose:
Free storage associated with a buffer.

Parameters:

buffer
The storage associated with the buffer will be freed.

Return value:
none

Synopsis:

```
void
gaa_free_uneval_cred (gaa_uneval_cred_ptr buffer /* IN */);
```

5.2.10. gaa_free_attribute routine

Purpose:
Free storage associated with a buffer.

Parameters:

buffer
The storage associated with the buffer will be freed.

Return value:
none

Synopsis:

```
void
gaa_free_attribute(gaa_attribute_ptr buffer /* IN */);
```

6. 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

7. Acknowledgments

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

8. Authors' Addresses

Tatyana Ryutov
Clifford Neuman
USC/Information Sciences Institute
4676 Admiralty Way Suite 1001

Marina del Rey, CA 90292-6695
Phone: +1 310 822 1511
E-Mail: {tryutov, bcn}@isi.edu